In [1…

```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import seaborn as sns
import numpy as np
from sklearn import preprocessing
from sklearn import ensemble
import warnings
warnings.filterwarnings("ignore")
```

# 1. Importing Data

In [2…

```
df = pd.read_csv("customer_booking.csv", encoding="ISO-8859-1
df.tail(20)
```

Out[2]:

| | num_passengers | sales_channel | trip_type | purchase_lead | length_ |
|---|---|---|---|---|---|
| **49980** | 4 | Internet | RoundTrip | 242 | |
| **49981** | 1 | Internet | RoundTrip | 317 | |
| **49982** | 2 | Internet | RoundTrip | 177 | |
| **49983** | 1 | Internet | RoundTrip | 112 | |
| **49984** | 2 | Internet | RoundTrip | 7 | |
| **49985** | 1 | Internet | RoundTrip | 26 | |
| **49986** | 1 | Internet | RoundTrip | 94 | |
| **49987** | 3 | Internet | RoundTrip | 243 | |
| **49988** | 1 | Internet | RoundTrip | 6 | |
| **49989** | 1 | Internet | RoundTrip | 33 | |
| **49990** | 1 | Internet | RoundTrip | 12 | |
| **49991** | 1 | Internet | RoundTrip | 8 | |
| **49992** | 1 | Internet | RoundTrip | 14 | |
| **49993** | 1 | Internet | RoundTrip | 19 | |
| **49994** | 2 | Internet | RoundTrip | 25 | |
| **49995** | 2 | Internet | RoundTrip | 27 | |
| **49996** | 1 | Internet | RoundTrip | 111 | |
| **49997** | 1 | Internet | RoundTrip | 24 | |
| **49998** | 1 | Internet | RoundTrip | 15 | |
| **49999** | 1 | Internet | RoundTrip | 19 | |

In [3…

```
df_without_index_header = df.copy()
df_without_index_header.reset_index(drop=True, inplace=True)
df_str = df_without_index_header.to_string(index=False, heade
```

In [4…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   num_passengers         50000 non-null  int64  
 1   sales_channel          50000 non-null  object 
 2   trip_type              50000 non-null  object 
 3   purchase_lead          50000 non-null  int64  
 4   length_of_stay         50000 non-null  int64  
 5   flight_hour            50000 non-null  int64  
 6   flight_day             50000 non-null  object 
 7   route                  50000 non-null  object 
 8   booking_origin         50000 non-null  object 
 9   wants_extra_baggage    50000 non-null  int64  
 10  wants_preferred_seat   50000 non-null  int64  
 11  wants_in_flight_meals  50000 non-null  int64  
 12  flight_duration        50000 non-null  float64
 13  booking_complete       50000 non-null  int64  
dtypes: float64(1), int64(8), object(5)
memory usage: 5.3+ MB
```

In [5…

```
print(df.columns)
```

```
Index(['num_passengers', 'sales_channel', 'trip_type', 'purc
hase_lead',
       'length_of_stay', 'flight_hour', 'flight_day', 'rout
e',
       'booking_origin', 'wants_extra_baggage', 'wants_prefe
rred_seat',
       'wants_in_flight_meals', 'flight_duration', 'booking_
complete'],
      dtype='object')
```

In [6…

```
df.describe()
```

Out[6]:

| | num_passengers | purchase_lead | length_of_stay | flight_hour | wan |
|---|---|---|---|---|---|
| **count** | 50000.000000 | 50000.000000 | 50000.00000 | 50000.00000 | |
| **mean** | 1.591240 | 84.940480 | 23.04456 | 9.06634 | |
| **std** | 1.020165 | 90.451378 | 33.88767 | 5.41266 | |
| **min** | 1.000000 | 0.000000 | 0.00000 | 0.00000 | |
| **25%** | 1.000000 | 21.000000 | 5.00000 | 5.00000 | |
| **50%** | 1.000000 | 51.000000 | 17.00000 | 9.00000 | |
| **75%** | 2.000000 | 115.000000 | 28.00000 | 13.00000 | |
| **max** | 9.000000 | 867.000000 | 778.00000 | 23.00000 | |

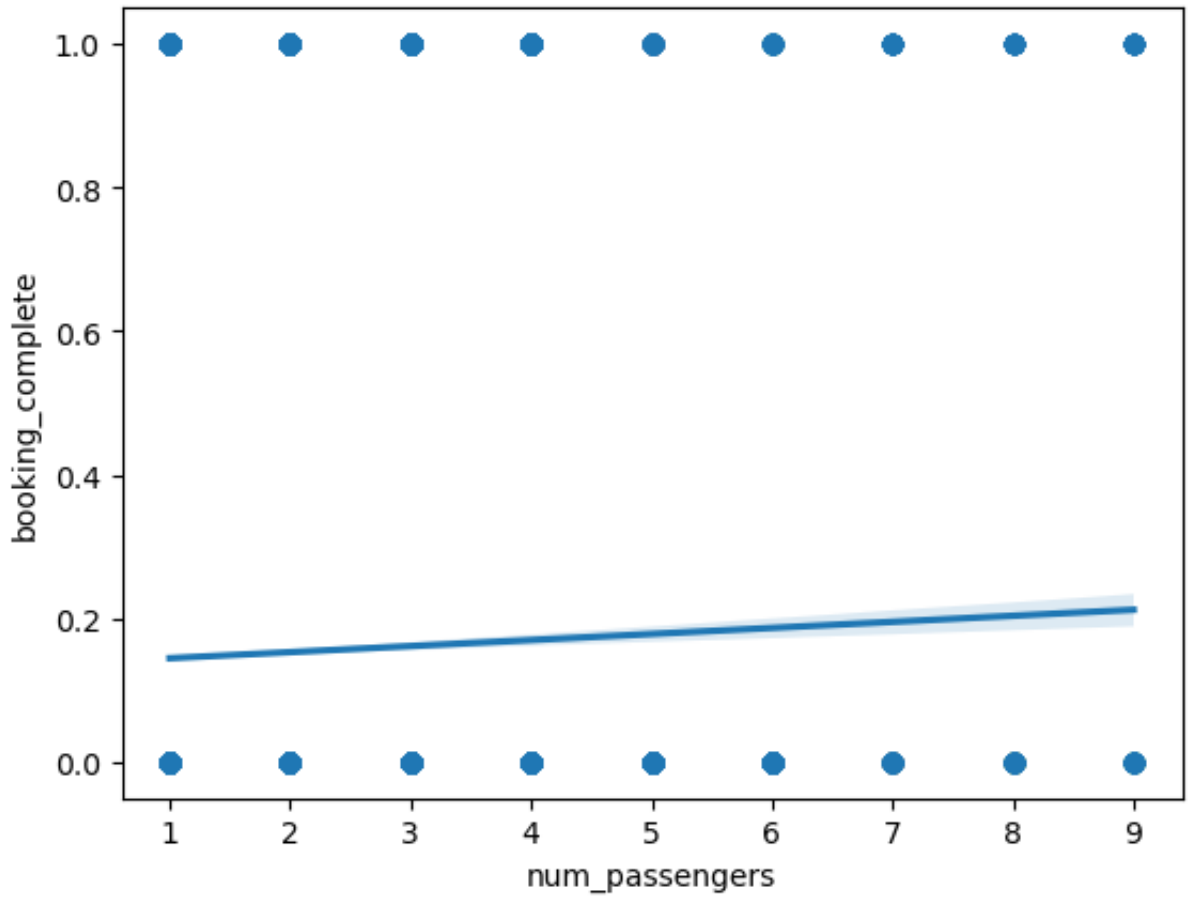
# 2. Data Preparation And Processing

In [7…

```
sns.regplot(x="num_passengers", y="booking_complete", data=df
```

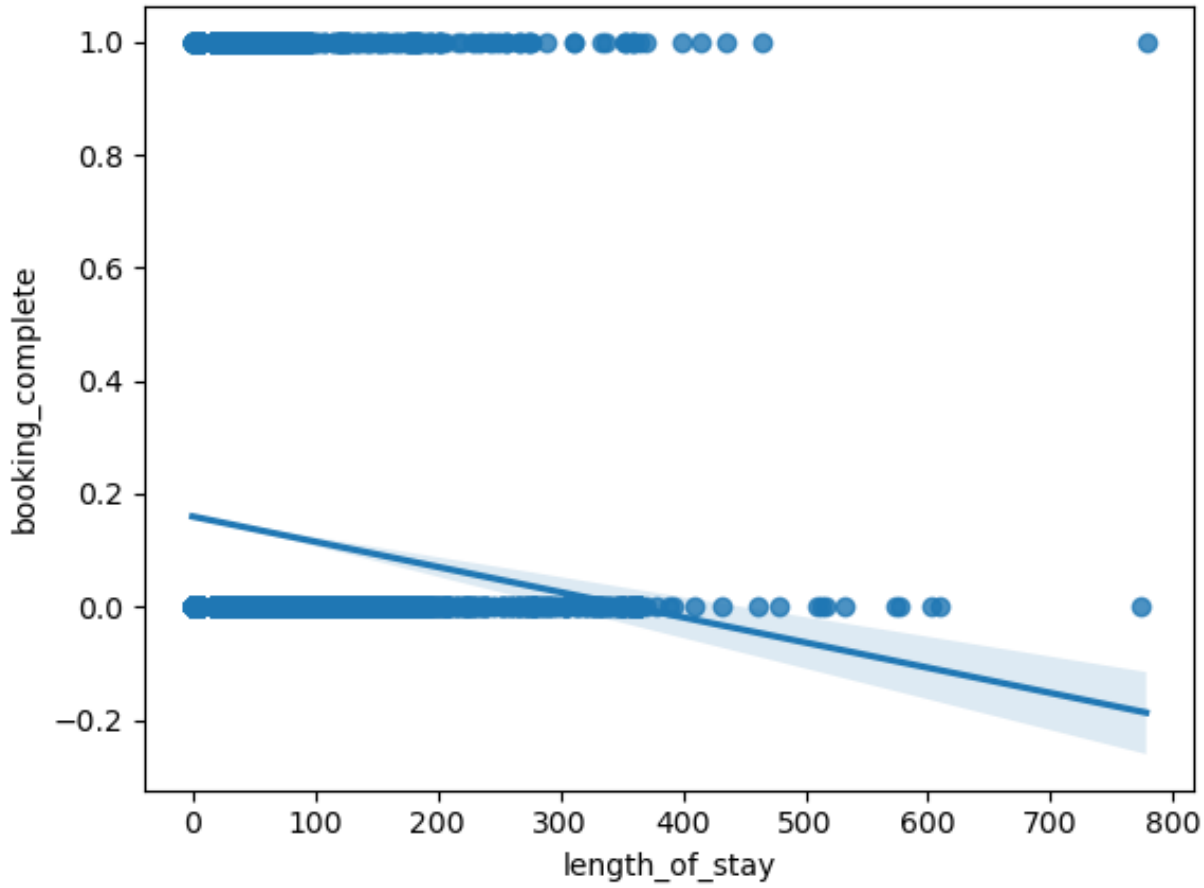
Out[7]:

```
<Axes: xlabel='num_passengers', ylabel='booking_complete'>
```



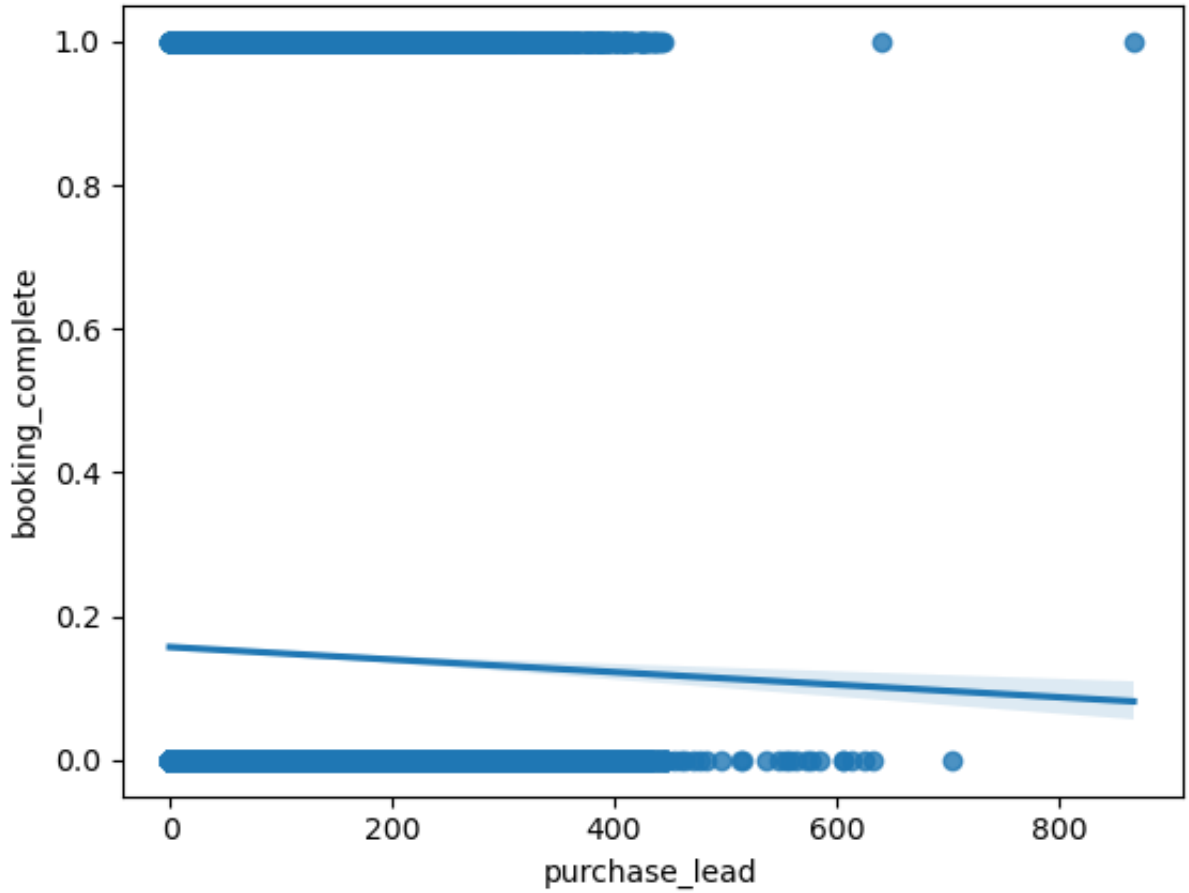
In [8…
```
sns.regplot(x="length_of_stay", y="booking_complete", data=df
```

Out[8]:
```
<Axes: xlabel='length_of_stay', ylabel='booking_complete'>
```




In [9…
```
sns.regplot(x="purchase_lead", y="booking_complete", data=df)
```

Out[9]:
```
<Axes: xlabel='purchase_lead', ylabel='booking_complete'>
```




## Feature Transformation

In [1…

```
Features= df[['num_passengers', 'sales_channel', 'trip_type',
       'length_of_stay', 'flight_day', 'wants_extra_baggage',
       'wants_in_flight_meals', 'flight_duration']].values
Features[0:5]
```

Out[10]:

```
array([[2, 'Internet', 'RoundTrip', 262, 19, 'Sat', 1, 0,
0, 5.52],
       [1, 'Internet', 'RoundTrip', 112, 20, 'Sat', 0, 0,
0, 5.52],
       [2, 'Internet', 'RoundTrip', 243, 22, 'Wed', 1, 1,
0, 5.52],
       [1, 'Internet', 'RoundTrip', 96, 31, 'Sat', 0, 0, 1,
5.52],
       [2, 'Internet', 'RoundTrip', 68, 22, 'Wed', 1, 0, 1,
5.52]],
      dtype=object)
```

In [1…

```
df.head(10)
```

Out[11]:

| | num_passengers | sales_channel | trip_type | purchase_lead | length_of_s |
|---|---|---|---|---|---|
| **0** | 2 | Internet | RoundTrip | 262 | |
| **1** | 1 | Internet | RoundTrip | 112 | |
| **2** | 2 | Internet | RoundTrip | 243 | |
| **3** | 1 | Internet | RoundTrip | 96 | |
| **4** | 2 | Internet | RoundTrip | 68 | |
| **5** | 1 | Internet | RoundTrip | 3 | |
| **6** | 3 | Internet | RoundTrip | 201 | |
| **7** | 2 | Internet | RoundTrip | 238 | |
| **8** | 1 | Internet | RoundTrip | 80 | |
| **9** | 1 | Mobile | RoundTrip | 378 | |

In [1…
```
df['sales_channel'].unique()
```

Out[12]:
```
array(['Internet', 'Mobile'], dtype=object)
```

In [1…
```
df['num_passengers'].unique()
```

Out[13]:
```
array([2, 1, 3, 4, 6, 5, 7, 9, 8])
```

In [1…
```
df['trip_type'].unique()
```

Out[14]:
```
array(['RoundTrip', 'CircleTrip', 'OneWay'], dtype=object)
```

In [1…
```
df['flight_day'].unique()
```

Out[15]:
```
array(['Sat', 'Wed', 'Thu', 'Mon', 'Sun', 'Tue', 'Fri'], dt
ype=object)
```

In [1…
```
from sklearn.preprocessing import OneHotEncoder

ohe=OneHotEncoder()
ohe.fit_transform(df[['sales_channel','trip_type','flight_day
```

Out[16]:
```
array([[1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 1.],
       ...,
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 1., 0., 0.]])
```

In [ …

In [1…
```
df_final=df
df_final
```

Out[17]:

| | num_passengers | sales_channel | trip_type | purchase_lead | length |
|---|---|---|---|---|---|
| **0** | 2 | Internet | RoundTrip | 262 | |
| **1** | 1 | Internet | RoundTrip | 112 | |
| **2** | 2 | Internet | RoundTrip | 243 | |
| **3** | 1 | Internet | RoundTrip | 96 | |
| **4** | 2 | Internet | RoundTrip | 68 | |
| **...** | ... | ... | ... | ... | |
| **49995** | 2 | Internet | RoundTrip | 27 | |
| **49996** | 1 | Internet | RoundTrip | 111 | |
| **49997** | 1 | Internet | RoundTrip | 24 | |
| **49998** | 1 | Internet | RoundTrip | 15 | |
| **49999** | 1 | Internet | RoundTrip | 19 | |

50000 rows × 14 columns

In [1…
```
feature_arry=ohe.fit_transform(df_final[['sales_channel','tri
ohe.categories_
```

Out[18]:
```
[array(['Internet', 'Mobile'], dtype=object),
 array(['CircleTrip', 'OneWay', 'RoundTrip'], dtype=objec
t),
 array(['Fri', 'Mon', 'Sat', 'Sun', 'Thu', 'Tue', 'Wed'], d
type=object)]
```

In [1…
```
feature_df = pd.DataFrame(feature_arry, columns=ohe.get_featu
feature_df
```

Out[19]:

| | sales_channel_Internet | sales_channel_Mobile | trip_type_CircleTrip |
|---|---|---|---|
| **0** | 1.0 | 0.0 | 0.0 |
| **1** | 1.0 | 0.0 | 0.0 |
| **2** | 1.0 | 0.0 | 0.0 |
| **3** | 1.0 | 0.0 | 0.0 |
| **4** | 1.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... |
| **49995** | 1.0 | 0.0 | 0.0 |
| **49996** | 1.0 | 0.0 | 0.0 |
| **49997** | 1.0 | 0.0 | 0.0 |
| **49998** | 1.0 | 0.0 | 0.0 |
| **49999** | 1.0 | 0.0 | 0.0 |

50000 rows × 12 columns

In [2…
```
feature_df.describe()
```

Out[20]:

| | sales_channel_Internet | sales_channel_Mobile | trip_type_CircleTrip |
|---|---|---|---|
| **count** | 50000.000000 | 50000.000000 | 50000.000000 |
| **mean** | 0.887640 | 0.112360 | 0.002320 |
| **std** | 0.315812 | 0.315812 | 0.048111 |
| **min** | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 1.000000 | 0.000000 | 0.000000 |
| **50%** | 1.000000 | 0.000000 | 0.000000 |
| **75%** | 1.000000 | 0.000000 | 0.000000 |
| **max** | 1.000000 | 1.000000 | 1.000000 |

In [2…
```
df_final = df_final.drop(['sales_channel','trip_type','flight
df_final
```

Out[21]:

| | num_passengers | purchase_lead | length_of_stay | flight_hour | war |
|---|---|---|---|---|---|
| **0** | 2 | 262 | 19 | 7 | |
| **1** | 1 | 112 | 20 | 3 | |
| **2** | 2 | 243 | 22 | 17 | |
| **3** | 1 | 96 | 31 | 4 | |
| **4** | 2 | 68 | 22 | 15 | |
| **...** | ... | ... | ... | ... | |
| **49995** | 2 | 27 | 6 | 9 | |
| **49996** | 1 | 111 | 6 | 4 | |
| **49997** | 1 | 24 | 6 | 22 | |
| **49998** | 1 | 15 | 6 | 11 | |
| **49999** | 1 | 19 | 6 | 10 | |

50000 rows × 9 columns

In [2…
```
df_final = pd.get_dummies(df_final, columns=['num_passengers'
```

In [2…
```
df_final.rename(columns={'booking_complete': 'label'}, inplac
```

In [2…
```
df_final= pd.concat([df_final, feature_df], axis=1)
df_final
```

Out[24]:

| | purchase_lead | length_of_stay | flight_hour | wants_extra_baggage |
|---|---|---|---|---|
| **0** | 262 | 19 | 7 | 1 |
| **1** | 112 | 20 | 3 | 0 |
| **2** | 243 | 22 | 17 | 1 |
| **3** | 96 | 31 | 4 | 0 |
| **4** | 68 | 22 | 15 | 1 |
| **...** | ... | ... | ... | ... |
| **49995** | 27 | 6 | 9 | 1 |
| **49996** | 111 | 6 | 4 | 0 |
| **49997** | 24 | 6 | 22 | 0 |
| **49998** | 15 | 6 | 11 | 1 |
| **49999** | 19 | 6 | 10 | 0 |

50000 rows × 29 columns

In [2…
```
df_final.describe()
```

Out[25]:

| | purchase_lead | length_of_stay | flight_hour | wants_extra_baggage |
|---|---|---|---|---|
| **count** | 50000.000000 | 50000.00000 | 50000.00000 | 50000.000000 |
| **mean** | 84.940480 | 23.04456 | 9.06634 | 0.668780 |
| **std** | 90.451378 | 33.88767 | 5.41266 | 0.470657 |
| **min** | 0.000000 | 0.00000 | 0.00000 | 0.000000 |
| **25%** | 21.000000 | 5.00000 | 5.00000 | 0.000000 |
| **50%** | 51.000000 | 17.00000 | 9.00000 | 1.000000 |
| **75%** | 115.000000 | 28.00000 | 13.00000 | 1.000000 |
| **max** | 867.000000 | 778.00000 | 23.00000 | 1.000000 |

## Reviewing Booking Probabilty from Total Data

In [2…
```
label=df[['booking_complete']]
```

In [2…
```
ohe=OneHotEncoder()
features_arry=ohe.fit_transform(df[['booking_complete']]).toa
booking_complete = pd.DataFrame(features_arry, columns=ohe.ge

booking_complete
```

Out[27]:

| | booking_complete_0 | booking_complete_1 |
|---|---|---|
| **0** | 1.0 | 0.0 |
| **1** | 1.0 | 0.0 |
| **2** | 1.0 | 0.0 |
| **3** | 1.0 | 0.0 |
| **4** | 1.0 | 0.0 |
| **...** | ... | ... |
| **49995** | 1.0 | 0.0 |
| **49996** | 1.0 | 0.0 |
| **49997** | 1.0 | 0.0 |
| **49998** | 1.0 | 0.0 |
| **49999** | 1.0 | 0.0 |

50000 rows × 2 columns

In [2…
```
zeros_count = (df['booking_complete'] == 0).sum()
ones_count = (df['booking_complete'] == 1).sum()
print(f'Total number of zeros in the "booking_complete" colum
print(f'Total number of ones in the "booking_complete" columr
```

```
Total number of zeros in the "booking_complete" column: 4252
2
Total number of ones in the "booking_complete" column: 7478
```

In [2…
```
Booking=booking_complete.transpose()
Booking
```

Out[29]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 49 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **booking_complete_0** | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | ... | |
| **booking_complete_1** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | |

2 rows × 50000 columns

In [3…

```
Booking['total']='42522','7478'
Booking
```

Out[30]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 49 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **booking_complete_0** | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | ... | |
| **booking_complete_1** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | |

2 rows × 50001 columns

In [3…

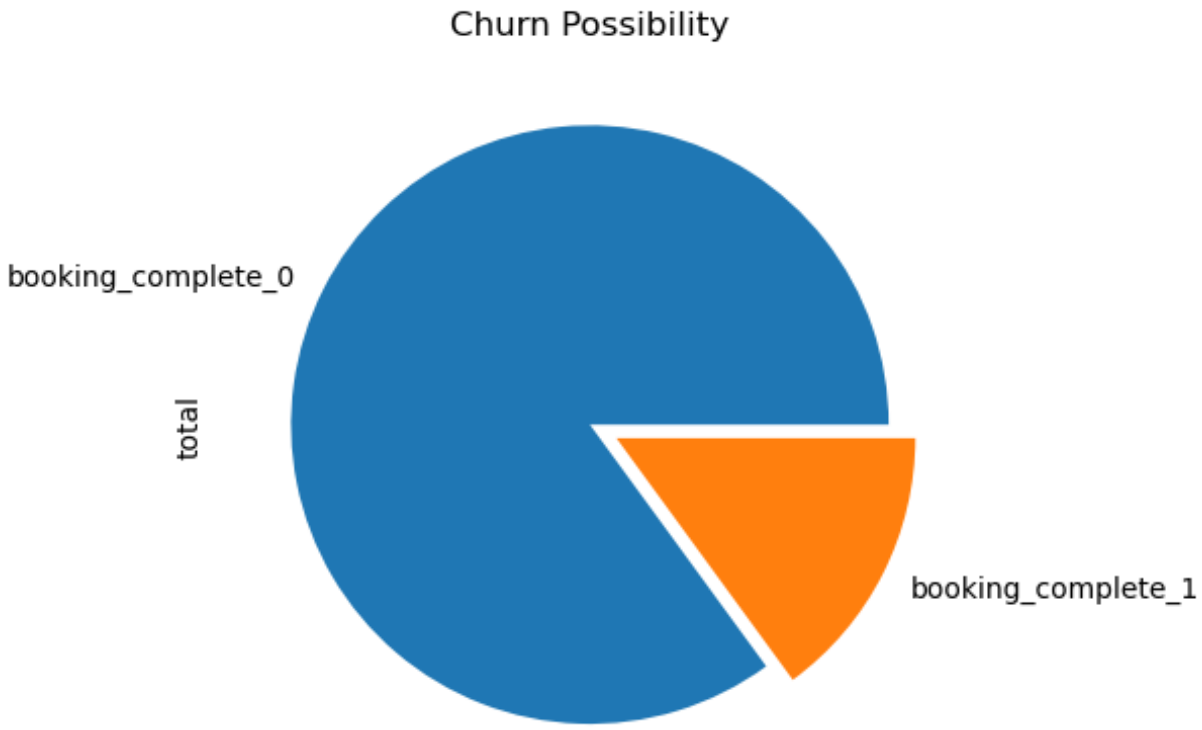
```
import plotly.graph_objects as go

plot_data=[
    go.Pie(
        labels=Booking.index,
        values=Booking['total'],
        marker=dict(colors=["Red","Green"],
                    line=dict(color="white",
                              width=1.5)),

        rotation=90,
        hoverinfo= 'label+value+text',
        hole=.6)
]


plot_layout = go.Layout(dict(title='Churn Possibility'))

fig = go.Figure(data=plot_data, layout=plot_layout)


fig.show()
```

## Churn Possibility

Completed bookings were 15% of the data

In [3…

```
Booking['total'] = pd.to_numeric(Booking['total'], errors='co


booking_complete_1 = [0, 0.1]
Booking['total'].plot(kind='pie', explode=booking_complete_1)
plt.title('Churn Possibility')
plt.show()
```

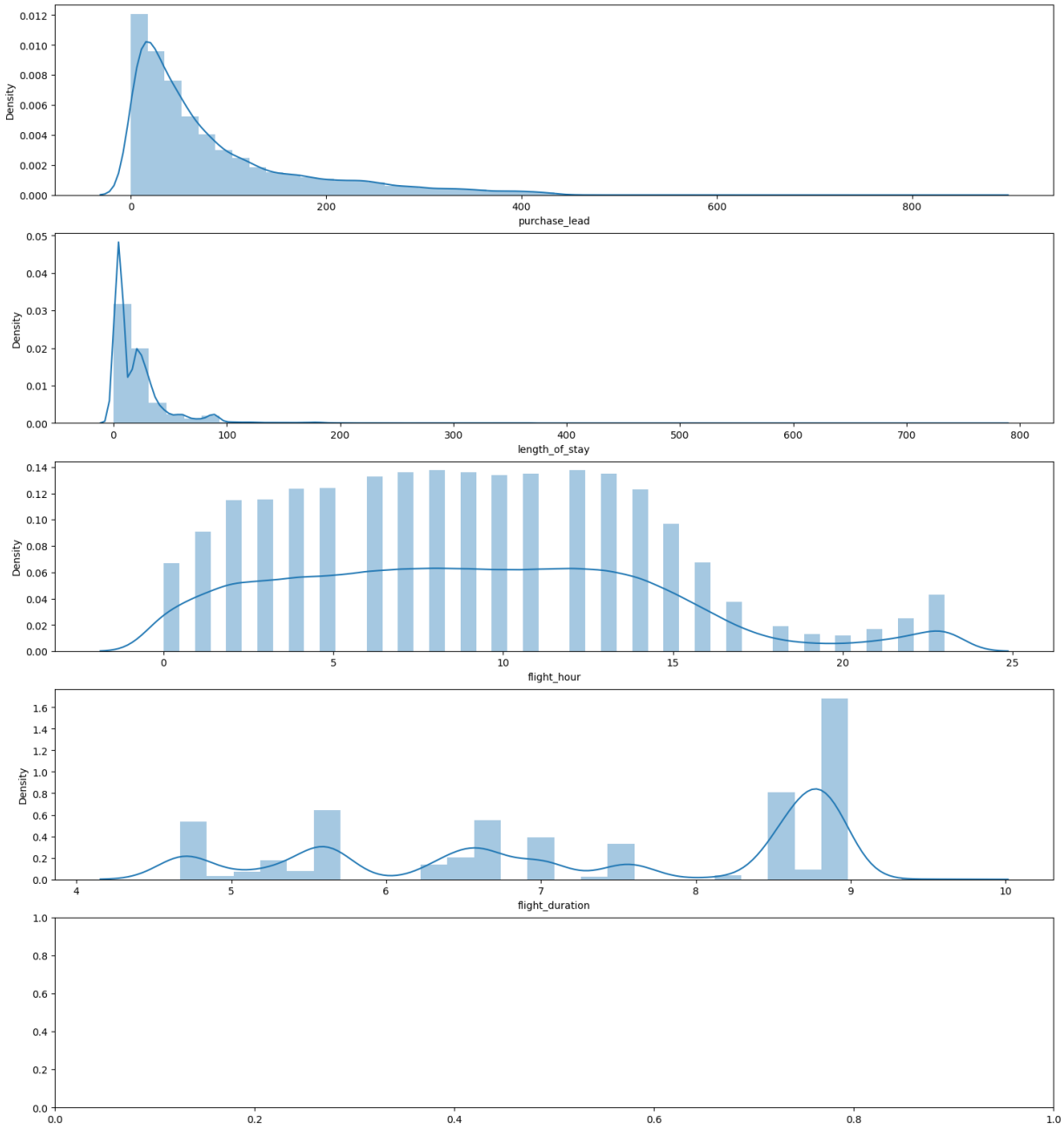



In [3…
```
df_final = df_final[[col for col in df_final.columns if col !
```

In [ …
```

```

## purchase_lead, length_of_stay, flight_hour and flight_duration show some level of Skewness

In [3…
```
fig, axs = plt.subplots(nrows=5, figsize=(18, 20))
# Plot histograms
sns.distplot((df_final["purchase_lead"].dropna()), ax=axs[0])
sns.distplot((df_final["length_of_stay"].dropna()), ax=axs[1]
sns.distplot((df_final["flight_hour"].dropna()), ax=axs[2])
sns.distplot((df_final["flight_duration"].dropna()), ax=axs[3
plt.show()
```

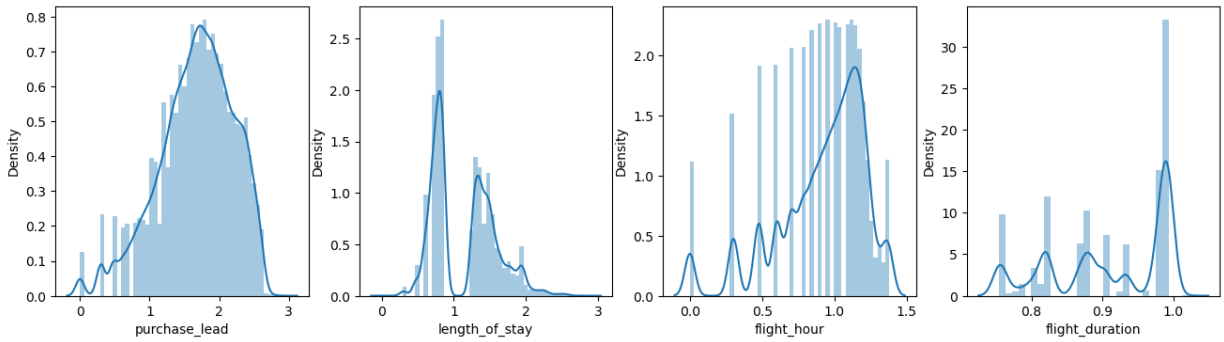



# Correcting skewness In Data

In [3…

```
df_final["purchase_lead"] = np.log10(df_final["purchase_lead"
df_final["length_of_stay"] = np.log10(df_final["length_of_sta
df_final["flight_hour"] = np.log10(df_final["flight_hour"] +
df_final["flight_duration"] = np.log10(df_final["flight_durat
```

In [3…
```
fig, axs = plt.subplots(1, 4, figsize=(16, 4))
sns.distplot((df_final["purchase_lead"].dropna()), ax=axs[0])
sns.distplot((df_final["length_of_stay"].dropna()), ax=axs[1]
sns.distplot((df_final["flight_hour"].dropna()), ax=axs[2])
sns.distplot((df_final["flight_duration"].dropna()), ax=axs[3

plt.show()
```




# Review of Outliers

In [3…
```
df_final.describe()
```

Out[37]:

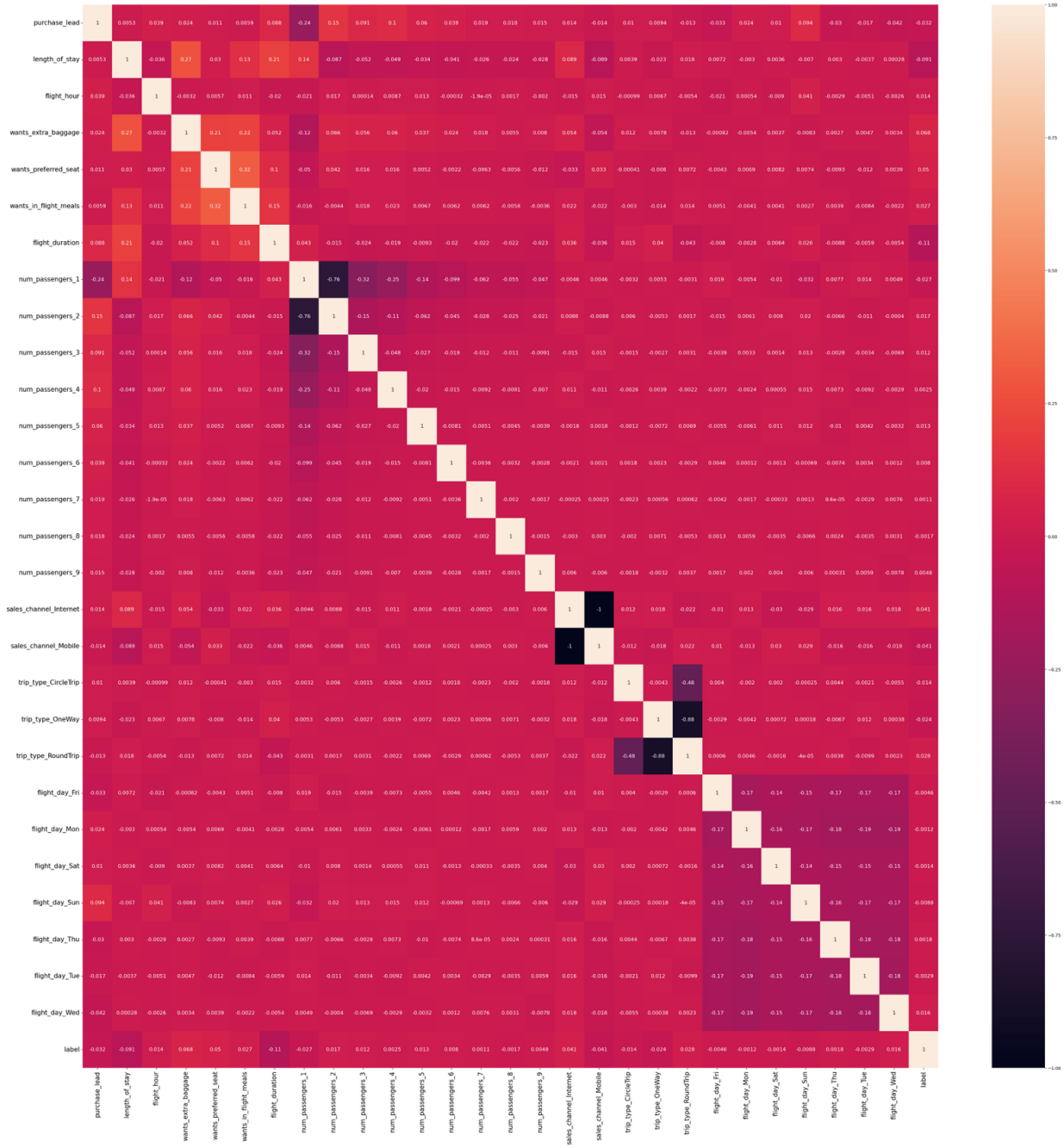
| | purchase_lead | length_of_stay | flight_hour | wants_extra_baggag |
|---|---|---|---|---|
| **count** | 50000.000000 | 50000.000000 | 50000.000000 | 50000.00000 |
| **mean** | 1.671739 | 1.141818 | 0.917859 | 0.66878 |
| **std** | 0.535219 | 0.436632 | 0.305857 | 0.47065 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.00000 |
| **25%** | 1.342423 | 0.778151 | 0.778151 | 0.00000 |
| **50%** | 1.716003 | 1.255273 | 1.000000 | 1.00000 |
| **75%** | 2.064458 | 1.462398 | 1.146128 | 1.00000 |
| **max** | 2.938520 | 2.891537 | 1.380211 | 1.00000 |

In [3…

```
import plotly.express as px
fig1=px.box(df_final, 'purchase_lead')
fig2=px.box(df_final, 'length_of_stay')
fig3=px.box(df_final, 'flight_hour')
fig4=px.box(df_final, 'flight_duration')
fig1.show()
fig2.show()
fig3.show()
fig4.show()
```

In [3…

```
correlation = df_final.corr()
plt.figure(figsize=(45, 45))
sns.heatmap(correlation, xticklabels=correlation.columns.valu
annot_kws={'size': 12}
)

plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```




In [4…
```
X =df_final.drop('label', axis=1)
Y=df_final['label']
```

In [4…
```
from sklearn.model_selection import train_test_split

X_train, X_tests, Y_train, Y_tests = train_test_split(X, Y, t
```

In [4…
```
from sklearn.metrics import accuracy_score
```

In [4…
```
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_tests.shape,  Y_tests.shape)
```

```
Train set: (40000, 28) (40000,)
Test set: (10000, 28) (10000,)
```

In [4…
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

In [4…
```
models=[LogisticRegression(), SVC(kernel='linear'),RandomFore
```

In [4…
```
def compare_models_train_test():
    for model in models:
        model.fit(X_train, Y_train)
        test_data_prediction=model.predict(X_tests)
        accuracy=accuracy_score(Y_tests, test_data_prediction

        print('Accuracy score of the ',model,'=',accuracy)
```

In [4…
```
compare_models_train_test()
```

```
Accuracy score of the  LogisticRegression() = 0.8504
Accuracy score of the  SVC(kernel='linear') = 0.8504
Accuracy score of the  RandomForestClassifier() = 0.8479
```

In [4…
```
from sklearn.model_selection import cross_val_score
```

In [4…
```
cv_score_lr=cross_val_score(LogisticRegression(max_iter=1000)
print('cv_score_lr=',cv_score_lr)

mean_accuracy_lr=sum(cv_score_lr)/len(cv_score_lr)
print('mean_accuracy_lr=',round(mean_accuracy_lr,2))
```

```
cv_score_lr= [0.8505 0.8505 0.8504 0.8503 0.8504]
mean_accuracy_lr= 0.85
```

In [ …

In [5…
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report, accuracy_s
from sklearn.inspection import permutation_importance

from yellowbrick.classifier import ConfusionMatrix
from sklearn.model_selection import GridSearchCV, RepeatedStr
```

In [5…
```
def model_fit_predict(model, X, Y, X_predict):
    model.fit(X,Y)
    return model.predict(X_predict)
def acc_score(Y_true, Y_pred):
    return accuracy_score(Y_true, Y_pred)
def pre_score(Y_true,Y_pred):
    return precision_score(Y_true, Y_pred)
def f_score(Y_true, Y_pred):
    return f1_score(Y_true, Y_pred)
```

In [5…
```
from sklearn.model_selection import train_test_split
X =df_final.drop('label', axis=1)
Y=df_final['label']
X_train, X_tests, Y_train, Y_tests = train_test_split(X, Y,te
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_tests.shape,  Y_tests.shape)
```

```
Train set: (40000, 28) (40000,)
Test set: (10000, 28) (10000,)
```

In [5…
```
model1 = RandomForestClassifier()

Y_pred_test = model_fit_predict(model1, X_train, Y_train, X_t


f1 = round(f1_score(Y_tests, Y_pred_test),2)


acc = round(accuracy_score(Y_tests, Y_pred_test),2)


pre = round(precision_score(Y_tests, Y_pred_test),2)

print(f"Accuracy, precision and f1-score for training data ar
```

```
Accuracy, precision and f1-score for training data are 0.85,
0.38 and 0.1 respectively
```

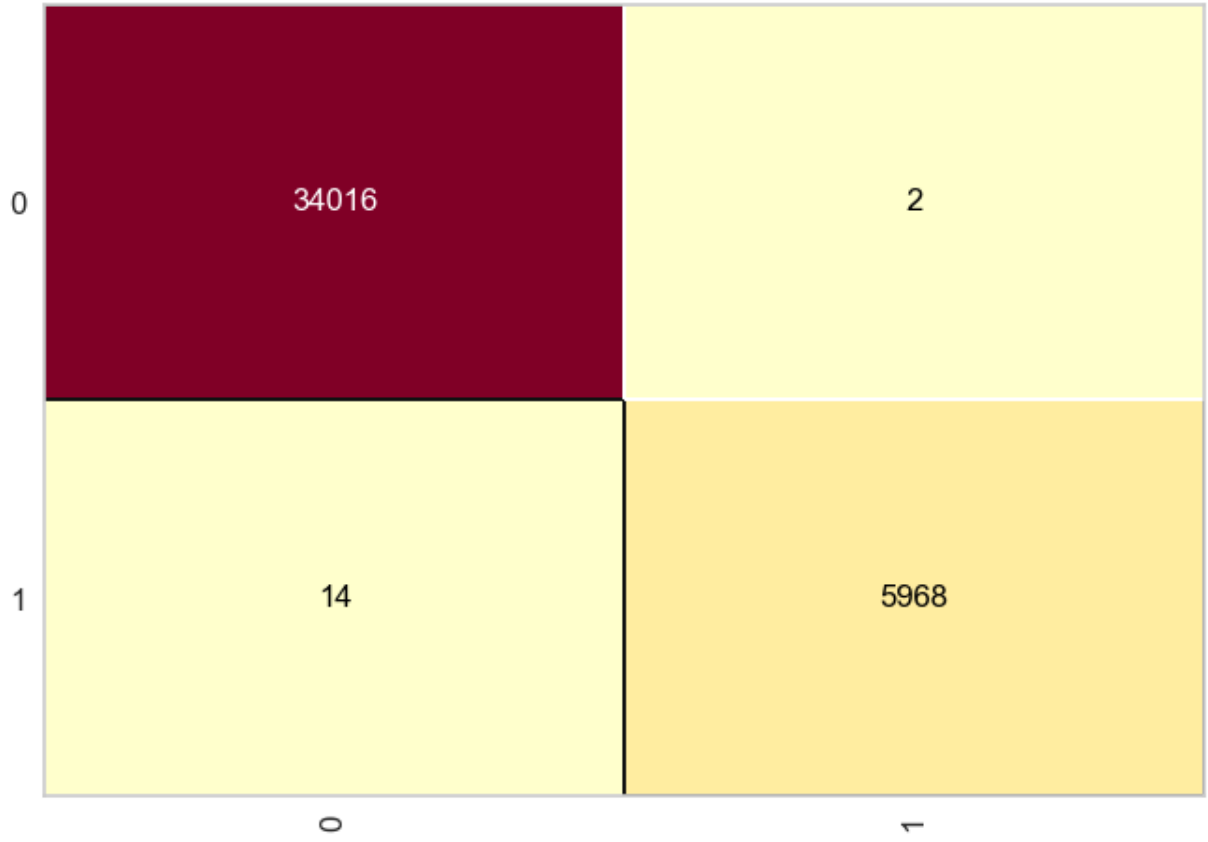
In [5…
```
from sklearn import metrics
predictions1 = model1.predict(X_tests)
tn, fp, fn, tp = metrics.confusion_matrix(Y_tests, prediction
Y_tests.value_counts()


print(f"True positives: {tp}")
print(f"False positives: {fp}")
print(f"True negatives: {tn}")
print(f"False negatives: {fn}\n")

print(f"Accuracy: {metrics.accuracy_score(Y_tests, prediction
print(f"Precision: {metrics.precision_score(Y_tests, predicti
print(f"Recall: {metrics.recall_score(Y_tests, predictions1)}
```

```
True positives: 83
False positives: 134
True negatives: 8370
False negatives: 1413

Accuracy: 0.8453
Precision: 0.3824884792626728
Recall: 0.05548128342245989
```

In [ …
```

```

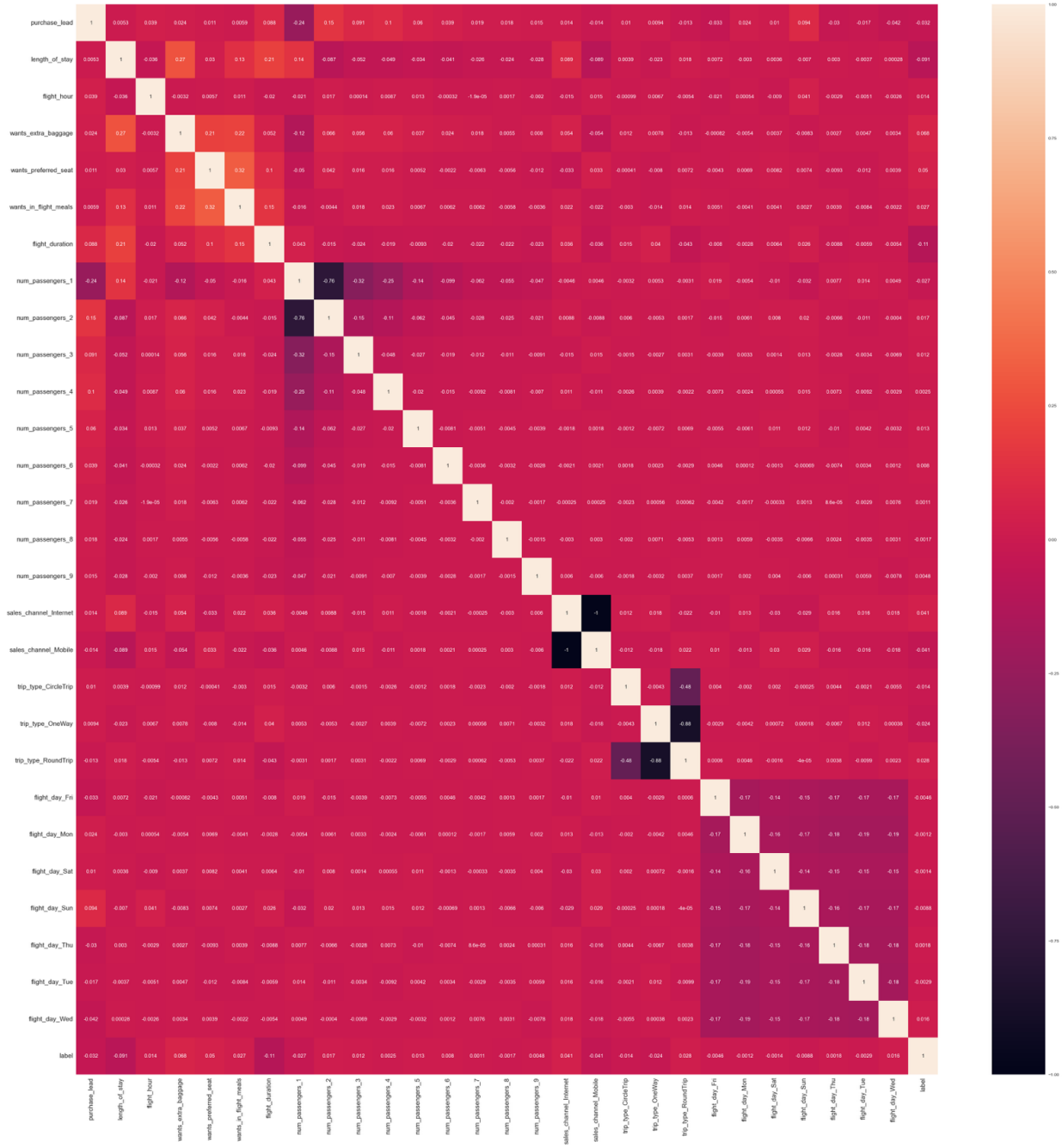
In [5…
```
cm= ConfusionMatrix(model1, classes=[0,1])
cm.fit(X_train, Y_train)

cm.score(X_train, Y_train)
```

Out[55]:
```
0.9996
```




In [5…

```
correlation = df_final.corr()
plt.figure(figsize=(45, 45))
sns.heatmap(
correlation, xticklabels=correlation.columns.values, yticklab
annot_kws={'size': 12}
)

plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```




In [5…

```
def correlation (dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range (len (corr_matrix.columns)) :
        for j in range(i):
            if abs (corr_matrix.iloc [i, j]) > threshold:
                colname = corr_matrix.columns [i]
                col_corr.add(colname)
    return col_corr
```

In [5…

```
corr_features=correlation(X_train, 0.85)
len(set(corr_features))
```

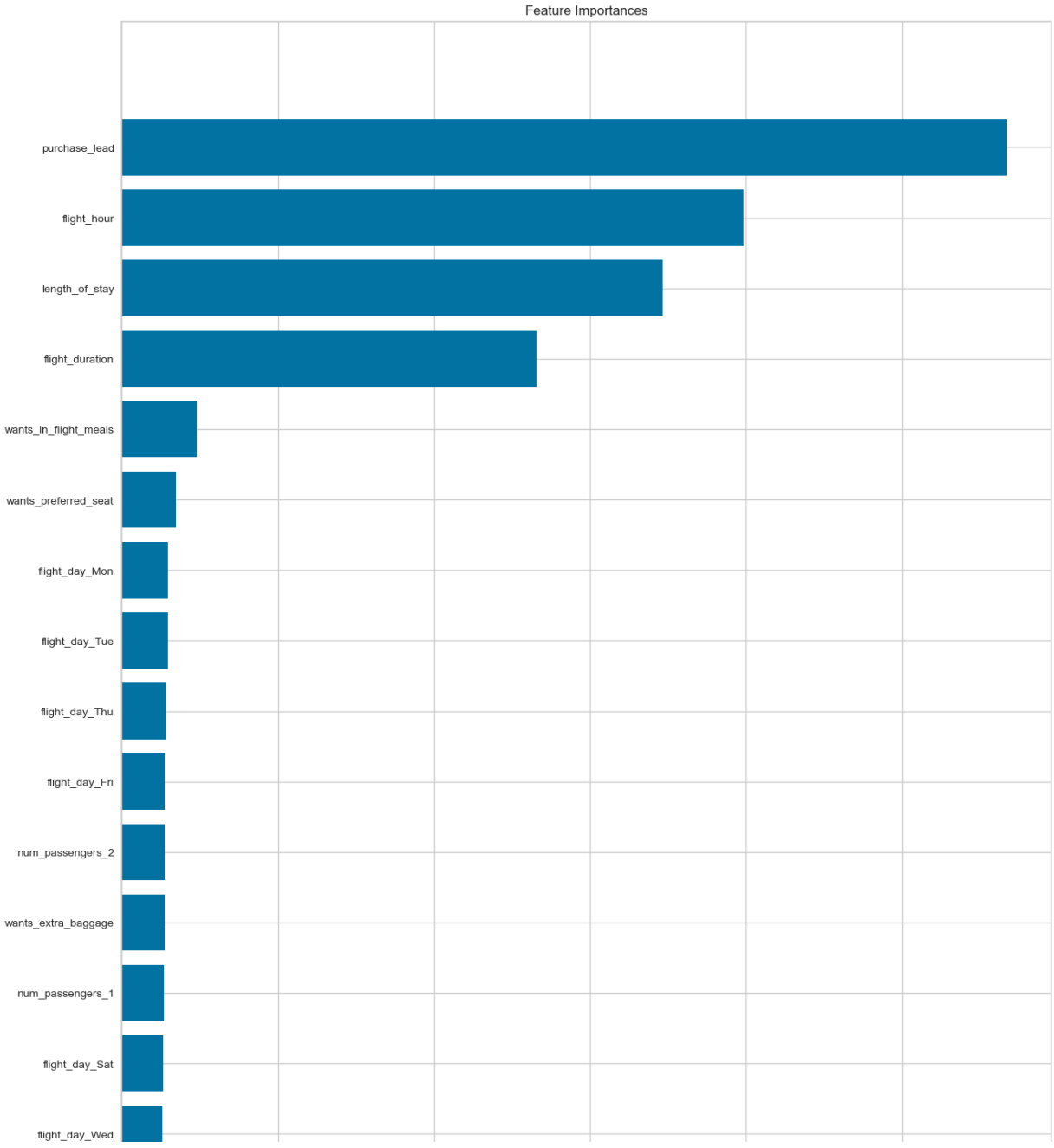
Out[58]: 2

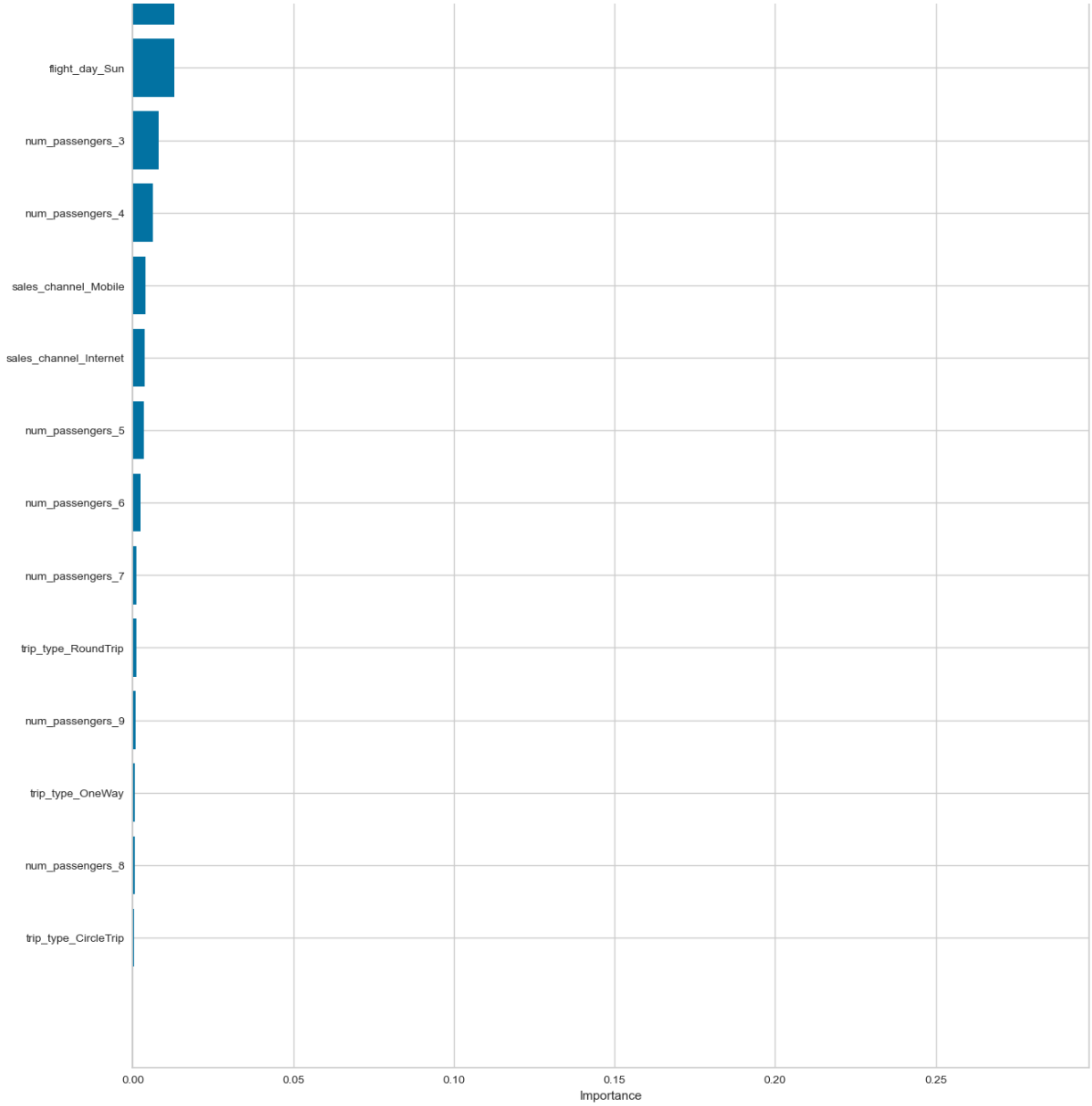
In [5…
```
corr_features
```

Out[59]: {'sales_channel_Mobile', 'trip_type_RoundTrip'}

In [6…
```
feature_importances = pd.DataFrame({'features': X_train.colum
```

In [6…
```
plt.figure(figsize=(15, 35))
plt.title('Feature Importances')
plt.barh(range(len(feature_importances)), feature_importances
plt.yticks(range(len(feature_importances)), feature_importanc
plt.xlabel('Importance')
plt.show()
```






In [6…

```
features_to_drop=['sales_channel_Mobile', 'trip_type_RoundTri
model_df=df_final.drop(columns=features_to_drop,axis=1)
model_df.head(5)
```

Out[62]:

| | purchase_lead | length_of_stay | flight_hour | wants_extra_baggage | wan |
|---|---|---|---|---|---|
| **0** | 2.419956 | 1.301030 | 0.903090 | 1 | |
| **1** | 2.053078 | 1.322219 | 0.602060 | 0 | |
| **2** | 2.387390 | 1.361728 | 1.255273 | 1 | |
| **3** | 1.986772 | 1.505150 | 0.698970 | 0 | |
| **4** | 1.838849 | 1.361728 | 1.204120 | 1 | |

5 rows × 27 columns

In [6…

```
X =model_df.drop('label', axis=1)
Y=model_df['label']
X_train, X_tests, Y_train, Y_tests = train_test_split(X, Y, t
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_tests.shape,  Y_tests.shape)
```

```
Train set: (40000, 26) (40000,)
Test set: (10000, 26) (10000,)
```

In [6…

```
#create an instance of the classifier and fit the training da
model2 = RandomForestClassifier()
Y_pred_test = model_fit_predict(model2, X_train, Y_train, X_t

#f1 score for training data
f1 = round(f1_score(Y_tests, Y_pred_test),3)

#accuracy score for training data
acc = round(accuracy_score(Y_tests, Y_pred_test),3)

recall=round(recall_score(Y_tests, Y_pred_test, pos_label=0),


#precision score for training data
pre = round(precision_score(Y_tests, Y_pred_test),3)

print(f"Accuracy, precision, recall and f1-score for training
```

```
Accuracy, precision, recall and f1-score for training data a
re 0.847, 0.406,0.987 and 0.092 respectively
```

In [6…
```
from sklearn import metrics

predictions2 = model2.predict(X_tests)
tn, fp, fn, tp = metrics.confusion_matrix(Y_tests, prediction
Y_tests.value_counts()


print(f"True positives: {tp}")
print(f"False positives: {fp}")
print(f"True negatives: {tn}")
print(f"False negatives: {fn}\n")

print(f"Accuracy: {metrics.accuracy_score(Y_tests, prediction
print(f"Precision: {metrics.precision_score(Y_tests, predicti
print(f"Recall: {metrics.recall_score(Y_tests, predictions2)}
```

```
True positives: 78
False positives: 114
True negatives: 8390
False negatives: 1418

Accuracy: 0.8468
Precision: 0.40625
Recall: 0.05213903743315508
```

In [6…
```
import numpy as np

max_features_range = np.arange(1,29,1)
n_estimators_range=np.arange(100,500,100)
param_grid=dict(max_features=max_features_range, n_estimators

rf=RandomForestClassifier()
grid=GridSearchCV(estimator=rf, param_grid=param_grid, cv=5 ,
```

In [6…
```
from sklearn.model_selection import GridSearchCV
```

In [6…

```
from sklearn.model_selection import GridSearchCV
classifier = ensemble.RandomForestClassifier(n_jobs=-1)
param_grid= {
    'n_estimators':[100,500,1000],
    'max_depth':[1,3,7,10],
    'criterion':["gini","entropy"],

}

model=GridSearchCV(
    estimator=classifier,
    param_grid=param_grid,
    scoring="accuracy",
    verbose=10,
    n_jobs=-1,
    cv=5,
)

model.fit(X_train,Y_train)
print(model.best_score_)
print(model.best_estimator_.get_params())
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fit
s
0.850475
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None,
'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqr
t', 'max_leaf_nodes': None, 'max_samples': None, 'min_impuri
ty_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100,
'n_jobs': -1, 'oob_score': False, 'random_state': None, 'ver
bose': 0, 'warm_start': False}
```

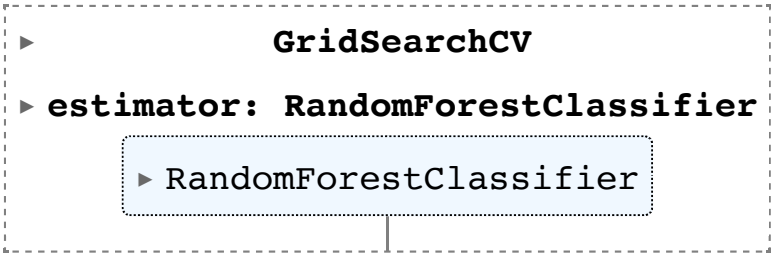
In [ …

In [6…
```
model3 = RandomForestClassifier(n_estimators= 500, n_jobs= -1

Y_pred_test = model_fit_predict(model3, X_train, Y_train, X_t

f1 = round(f1_score(Y_tests, Y_pred_test),2)
acc = round(accuracy_score(Y_tests, Y_pred_test),2)
recall=round(recall_score(Y_tests, Y_pred_test, pos_label=0),

pre = round(precision_score(Y_tests, Y_pred_test),2)
print(f"Accuracy, precision, recall and f1-score for training
```

```
Accuracy, precision, recall and f1-score for training data a
re 0.85, 0.4,0.99 and 0.09 respectively
```

In [7…
```
grid.fit(X_train,Y_train)
```

Out[70]:



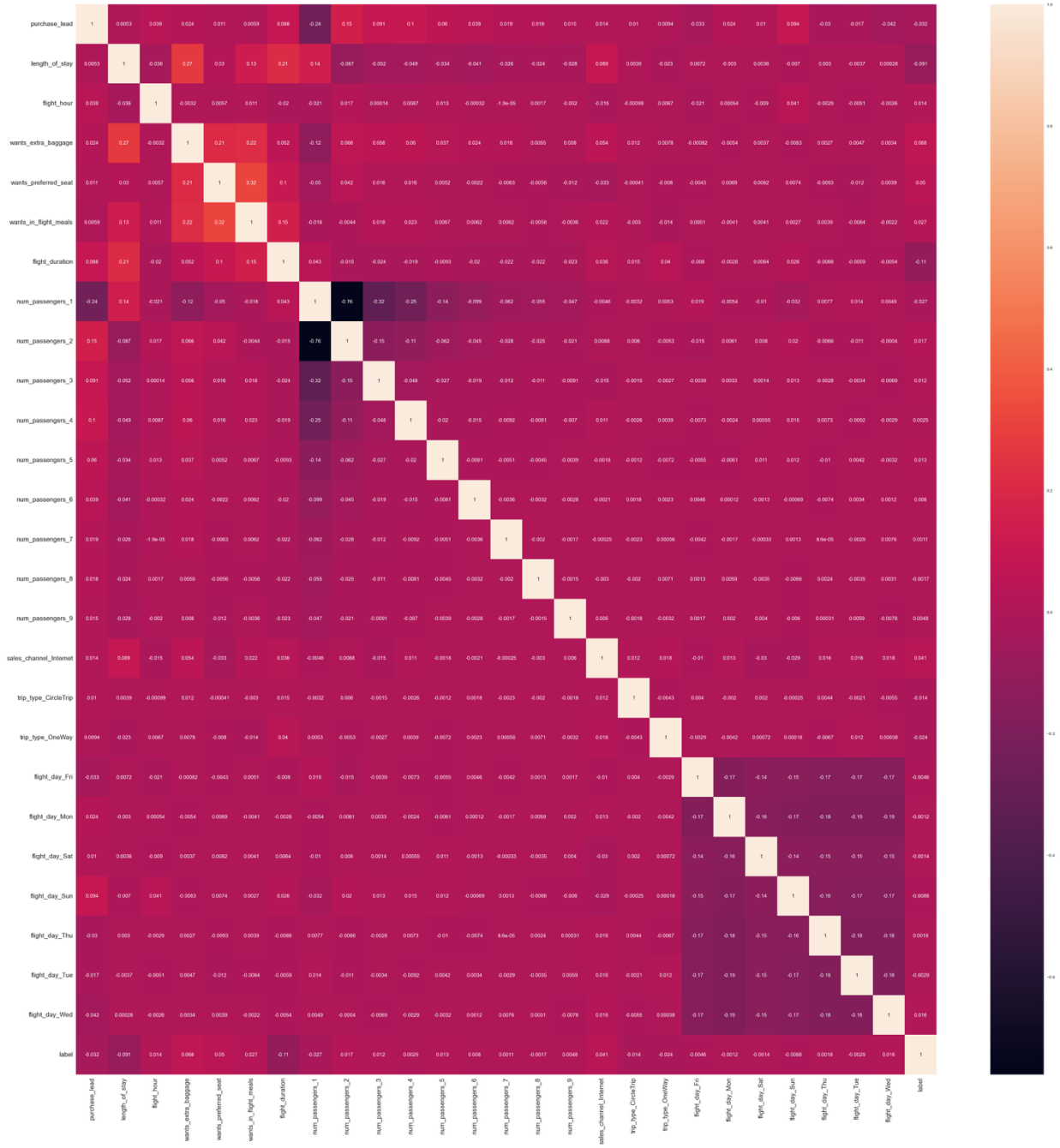

In [7…
```
print('The best Parameters are %s with a score of %0.2f'
      %(grid.best_params_, grid.best_score_))
```

```
The best Parameters are {'max_features': 6, 'n_estimators':
400} with a score of 0.85
```

# Verifying Parameters needed

In [7…

```
correlation = model_df.corr()
plt.figure(figsize=(45, 45))
sns.heatmap(
correlation, xticklabels=correlation.columns.values, yticklab
annot_kws={'size': 12}
)

plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



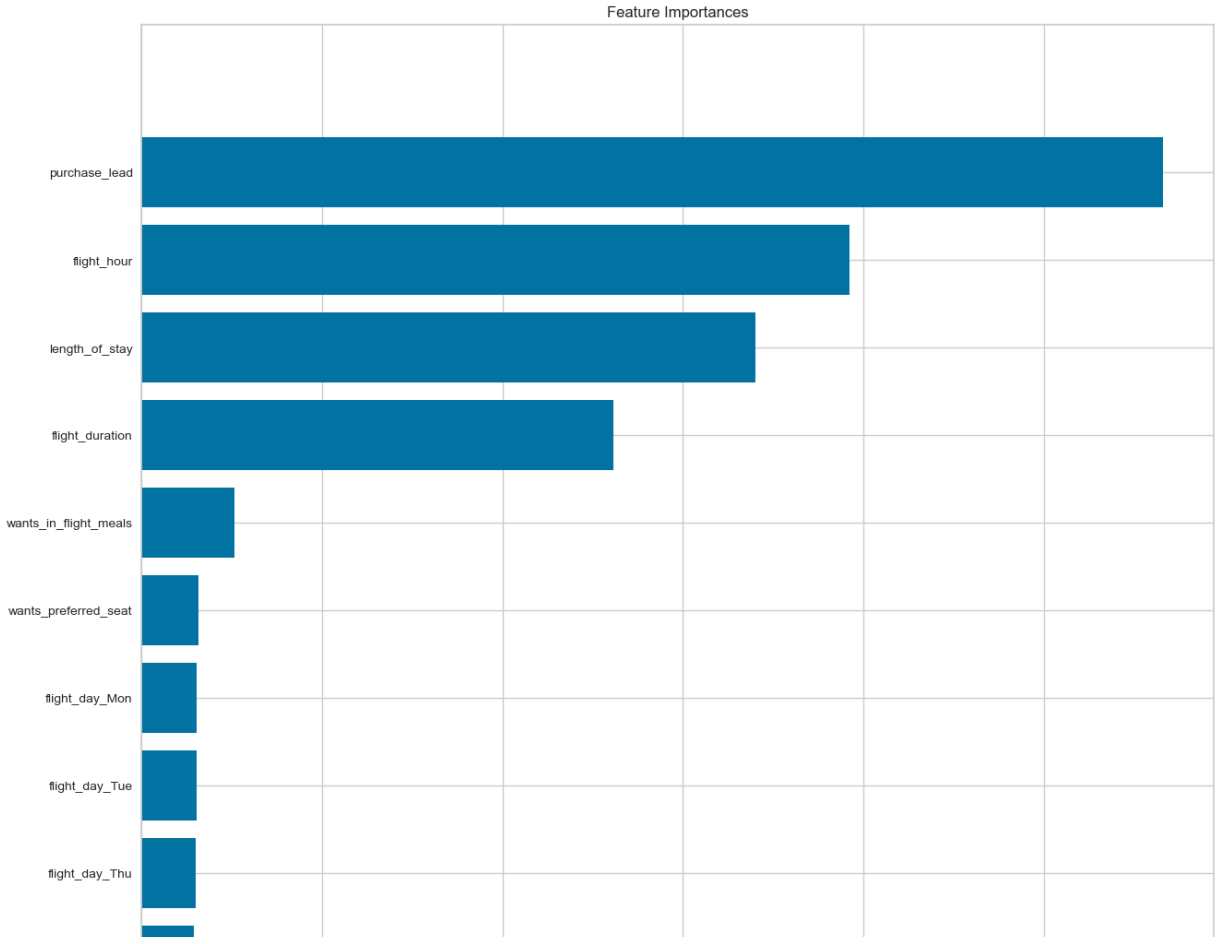
In [7…
```
def correlation (dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range (len (corr_matrix.columns)) :
        for j in range(i):
            if abs (corr_matrix.iloc [i, j]) > threshold:
                colname = corr_matrix.columns [i]
                col_corr.add(colname)
    return col_corr
```

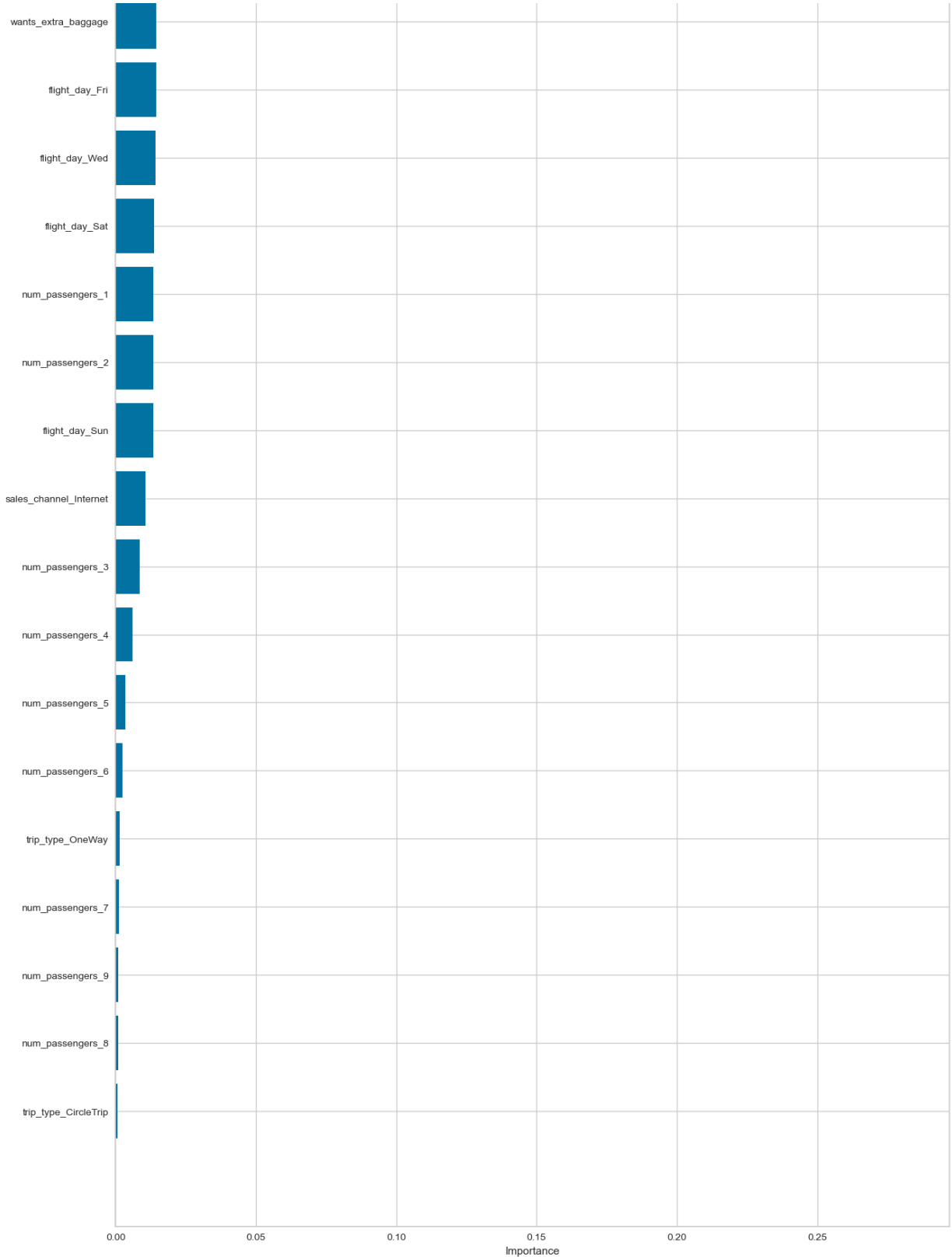
In [7…
```
corr_features=correlation(X_train, 0.7)
len(set(corr_features))
```

Out[74]: 1

In [7…
```
feature_importances = pd.DataFrame({'features': X_train.colum
```

In [7…
```
plt.figure(figsize=(15, 35))
plt.title('Feature Importances')
plt.barh(range(len(feature_importances)), feature_importances
plt.yticks(range(len(feature_importances)), feature_importanc
plt.xlabel('Importance')
plt.show()
```






# num_passengers,purchase_lead,length_of_stay,flight were the only features that showed corelation to completed booking

In [7…
```
from imblearn.under_sampling import RandomUnderSampler
```

In [7…
```
count_class_0, count_class_1=model_df.label.value_counts()
model_df_0=model_df[model_df['label']==0]
model_df_1=model_df[model_df['label']==1]
```

In [8…
```
model_df_0_under=model_df_0.sample(count_class_1)
SampledDf=pd.concat([model_df_0_under,model_df_1], axis=0)
SampledDf.head(5)
```

Out[80]:

| | purchase_lead | length_of_stay | flight_hour | wants_extra_baggage |
|---|---|---|---|---|
| **11778** | 1.568202 | 1.447158 | 0.301030 | 1 |
| **34542** | 2.152288 | 0.698970 | 0.845098 | 0 |
| **44778** | 1.944483 | 0.845098 | 0.301030 | 0 |
| **17798** | 1.913814 | 1.505150 | 1.041393 | 1 |
| **10186** | 1.812913 | 1.414973 | 1.230449 | 1 |

5 rows × 27 columns

In [ …
```

```

In [8…
```
Y = SampledDf['label']
X = SampledDf.drop(columns=['num_passengers_1', 'num_passenge
X_train, X_tests, Y_train, Y_tests = train_test_split(X, Y, t
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_tests.shape,  Y_tests.shape)
```

```
Train set: (10469, 24) (10469,)
Test set: (4487, 24) (4487,)
```

In [8…
```
model4 = RandomForestClassifier( n_estimators=100)
Y_pred_test = model_fit_predict(model4, X_train, Y_train, X_t


f1 =a round(f1_score(Y_tests, Y_pred_test),3)


acc = round(accuracy_score(Y_tests, Y_pred_test),3)

recall=round(recall_score(Y_tests, Y_pred_test, pos_label=0),


pre = round(precision_score(Y_tests, Y_pred_test),3)

print(f"Accuracy, precision, recall and f1-score for training
```

```
Accuracy, precision, recall and f1-score for training data a
re 0.61, 0.614,0.614 and 0.611 respectively
```

In [8…
```
from sklearn import metrics
predictions1 = model4.predict(X_tests)
tn, fp, fn, tp = metrics.confusion_matrix(Y_tests, prediction
Y_tests.value_counts()
```

Out[83]:
```
label
1    2257
0    2230
Name: count, dtype: int64
```

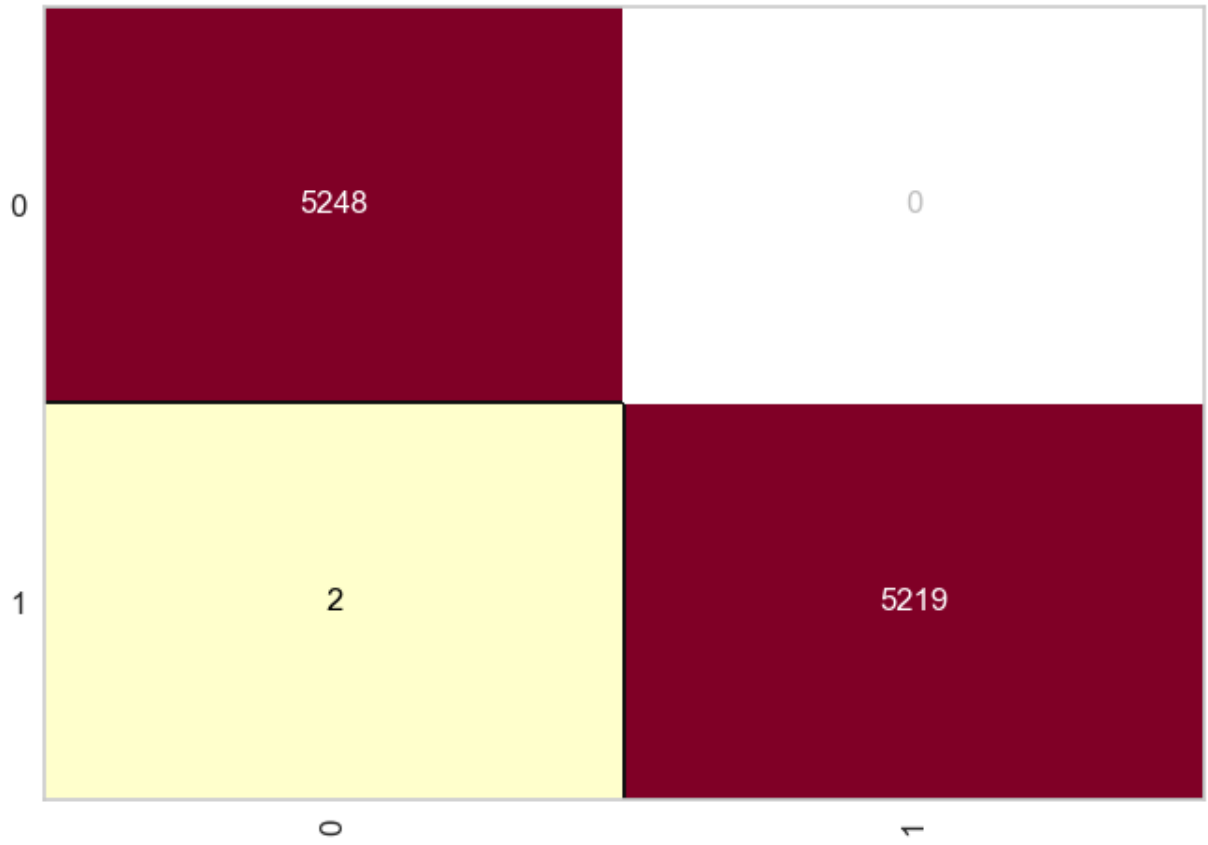
In [8…
```
print(f"True positives: {tp}")
print(f"False positives: {fp}")
print(f"True negatives: {tn}")
print(f"False negatives: {fn}\n")

print(f"Accuracy: {metrics.accuracy_score(Y_tests, prediction
print(f"Precision: {metrics.precision_score(Y_tests, predicti
print(f"Recall: {metrics.recall_score(Y_tests, predictions1)}
```

```
True positives: 1370
False positives: 861
True negatives: 1369
False negatives: 887

Accuracy: 0.6104301314909739
Precision: 0.6140744060959211
Recall: 0.6070004430660169
```

In [8…

```
cm= ConfusionMatrix(model4, classes=[0,1])
cm.fit(X_train, Y_train)

cm.score(X_train, Y_train)
```

Out[85]: 0.999808959786035



In [8…

```
from sklearn.model_selection import GridSearchCV
classifier = ensemble.RandomForestClassifier(n_jobs=-1)
param_grid= {
    'n_estimators':[100,500,1000],
    'max_depth':[1,3,7,10],
    'criterion':["gini","entropy"],

}

model=GridSearchCV(
    estimator=classifier,
    param_grid=param_grid,
    scoring="accuracy",
    verbose=10,
    n_jobs=-1,
    cv=5,
)

model.fit(X_train,Y_train)
print(model.best_score_)
print(model.best_estimator_.get_params())
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fit
s
0.6379797396241896
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None,
'criterion': 'entropy', 'max_depth': 7, 'max_features': 'sqr
t', 'max_leaf_nodes': None, 'max_samples': None, 'min_impuri
ty_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100
0, 'n_jobs': -1, 'oob_score': False, 'random_state': None, '
verbose': 0, 'warm_start': False}
[CV 1/5; 1/24] START criterion=gini, max_depth=1, n_estimato
rs=100.............
[CV 1/5; 1/24] END criterion=gini, max_depth=1, n_estimators
=100;, score=0.591 total time=   1.5s
[CV 1/5; 2/24] START criterion=gini, max_depth=1, n_estimato
rs=500.............
[CV 1/5; 2/24] END criterion=gini, max_depth=1, n_estimators
=500;, score=0.586 total time=   6.1s
[CV 5/5; 2/24] START criterion=gini, max_depth=1, n_estimato
rs=500.............
[CV 5/5; 2/24] END criterion=gini, max_depth=1, n_estimators
=500;, score=0.611 total time=   5.2s
[CV 4/5; 3/24] START criterion=gini, max_depth=1, n_estimato
rs=1000............
[CV 4/5; 3/24] END criterion=gini, max_depth=1, n_estimators
```

```
=1000;, score=0.632 total time=  13.1s
[CV 3/5; 5/24] START criterion=gini, max_depth=3, n_estimato
rs=500.............
[CV 3/5; 5/24] END criterion=gini, max_depth=3, n_estimators
=500;, score=0.629 total time=   7.4s
[CV 2/5; 6/24] START criterion=gini, max_depth=3, n_estimato
rs=1000............
[CV 2/5; 6/24] END criterion=gini, max_depth=3, n_estimators
=1000;, score=0.638 total time=  13.9s
[CV 1/5; 7/24] START criterion=gini, max_depth=7, n_estimato
rs=100.............
[CV 1/5; 7/24] END criterion=gini, max_depth=7, n_estimators
=100;, score=0.617 total time=   1.4s
[CV 2/5; 7/24] START criterion=gini, max_depth=7, n_estimato
rs=100.............
[CV 2/5; 7/24] END criterion=gini, max_depth=7, n_estimators
=100;, score=0.641 total time=   1.4s
[CV 3/5; 7/24] START criterion=gini, max_depth=7, n_estimato
rs=100.............
[CV 3/5; 7/24] END criterion=gini, max_depth=7, n_estimators
=100;, score=0.624 total time=   1.8s
[CV 1/5; 8/24] START criterion=gini, max_depth=7, n_estimato
rs=500.............
[CV 1/5; 8/24] END criterion=gini, max_depth=7, n_estimators
=500;, score=0.619 total time=   9.7s
[CV 5/5; 8/24] START criterion=gini, max_depth=7, n_estimato
rs=500.............
[CV 5/5; 8/24] END criterion=gini, max_depth=7, n_estimators
=500;, score=0.646 total time=   8.4s
[CV 4/5; 9/24] START criterion=gini, max_depth=7, n_estimato
rs=1000............
[CV 4/5; 9/24] END criterion=gini, max_depth=7, n_estimators
=1000;, score=0.649 total time=  21.7s
[CV 2/5; 11/24] START criterion=gini, max_depth=10, n_estima
tors=500...........
[CV 2/5; 11/24] END criterion=gini, max_depth=10, n_estimato
rs=500;, score=0.645 total time=  12.2s
[CV 1/5; 12/24] START criterion=gini, max_depth=10, n_estima
tors=1000..........
[CV 1/5; 12/24] END criterion=gini, max_depth=10, n_estimato
rs=1000;, score=0.617 total time=  24.2s
[CV 5/5; 12/24] START criterion=gini, max_depth=10, n_estima
tors=1000..........
[CV 5/5; 12/24] END criterion=gini, max_depth=10, n_estimato
rs=1000;, score=0.642 total time=  24.5s
[CV 4/5; 15/24] START criterion=entropy, max_depth=1, n_esti
mators=1000........
[CV 4/5; 15/24] END criterion=entropy, max_depth=1, n_estima
```

```
tors=1000;, score=0.633 total time=  13.8s
[CV 3/5; 17/24] START criterion=entropy, max_depth=3, n_esti
mators=500.........
[CV 3/5; 17/24] END criterion=entropy, max_depth=3, n_estima
tors=500;, score=0.628 total time=   7.3s
[CV 1/5; 18/24] START criterion=entropy, max_depth=3, n_esti
mators=1000.........
[CV 1/5; 18/24] END criterion=entropy, max_depth=3, n_estima
tors=1000;, score=0.603 total time=  15.9s
[CV 5/5; 18/24] START criterion=entropy, max_depth=3, n_esti
mators=1000.........
[CV 5/5; 18/24] END criterion=entropy, max_depth=3, n_estima
tors=1000;, score=0.626 total time=  15.0s
[CV 4/5; 20/24] START criterion=entropy, max_depth=7, n_esti
mators=500.........
[CV 4/5; 20/24] END criterion=entropy, max_depth=7, n_estima
tors=500;, score=0.647 total time=   9.8s
[CV 3/5; 21/24] START criterion=entropy, max_depth=7, n_esti
mators=1000.........
[CV 3/5; 21/24] END criterion=entropy, max_depth=7, n_estima
tors=1000;, score=0.632 total time=  20.3s
[CV 5/5; 22/24] START criterion=entropy, max_depth=10, n_est
imators=100.........
[CV 5/5; 22/24] END criterion=entropy, max_depth=10, n_estim
ators=100;, score=0.648 total time=   2.6s
[CV 3/5; 23/24] START criterion=entropy, max_depth=10, n_est
imators=500.........
[CV 3/5; 23/24] END criterion=entropy, max_depth=10, n_estim
ators=500;, score=0.629 total time=  13.3s
[CV 2/5; 24/24] START criterion=entropy, max_depth=10, n_est
imators=1000.........
[CV 2/5; 24/24] END criterion=entropy, max_depth=10, n_estim
ators=1000;, score=0.641 total time=  25.2s
[CV 3/5; 1/24] START criterion=gini, max_depth=1, n_estimato
rs=100.............
[CV 3/5; 1/24] END criterion=gini, max_depth=1, n_estimators
=100;, score=0.592 total time=   1.9s
[CV 3/5; 2/24] START criterion=gini, max_depth=1, n_estimato
rs=500.............
[CV 3/5; 2/24] END criterion=gini, max_depth=1, n_estimators
=500;, score=0.592 total time=   6.3s
[CV 2/5; 3/24] START criterion=gini, max_depth=1, n_estimato
rs=1000.............
[CV 2/5; 3/24] END criterion=gini, max_depth=1, n_estimators
=1000;, score=0.622 total time=  11.5s
[CV 5/5; 3/24] START criterion=gini, max_depth=1, n_estimato
rs=1000.............
[CV 5/5; 3/24] END criterion=gini, max_depth=1, n_estimators
```

```
=1000;, score=0.615 total time=  12.4s
[CV 1/5; 6/24] START criterion=gini, max_depth=3, n_estimato
rs=1000.............
[CV 1/5; 6/24] END criterion=gini, max_depth=3, n_estimators
=1000;, score=0.607 total time=  14.2s
[CV 5/5; 6/24] START criterion=gini, max_depth=3, n_estimato
rs=1000.............
[CV 5/5; 6/24] END criterion=gini, max_depth=3, n_estimators
=1000;, score=0.633 total time=  15.4s
[CV 4/5; 8/24] START criterion=gini, max_depth=7, n_estimato
rs=500..............
[CV 4/5; 8/24] END criterion=gini, max_depth=7, n_estimators
=500;, score=0.649 total time=   8.4s
[CV 3/5; 9/24] START criterion=gini, max_depth=7, n_estimato
rs=1000.............
[CV 3/5; 9/24] END criterion=gini, max_depth=7, n_estimators
=1000;, score=0.629 total time=  21.6s
[CV 1/5; 11/24] START criterion=gini, max_depth=10, n_estima
tors=500............
[CV 1/5; 11/24] END criterion=gini, max_depth=10, n_estimato
rs=500;, score=0.615 total time=  12.0s
[CV 5/5; 11/24] START criterion=gini, max_depth=10, n_estima
tors=500............
[CV 5/5; 11/24] END criterion=gini, max_depth=10, n_estimato
rs=500;, score=0.643 total time=  11.9s
[CV 4/5; 12/24] START criterion=gini, max_depth=10, n_estima
tors=1000...........
[CV 4/5; 12/24] END criterion=gini, max_depth=10, n_estimato
rs=1000;, score=0.652 total time=  24.7s
[CV 3/5; 14/24] START criterion=entropy, max_depth=1, n_esti
mators=500..........
[CV 3/5; 14/24] END criterion=entropy, max_depth=1, n_estima
tors=500;, score=0.592 total time=   7.4s
[CV 1/5; 15/24] START criterion=entropy, max_depth=1, n_esti
mators=1000.........
[CV 1/5; 15/24] END criterion=entropy, max_depth=1, n_estima
tors=1000;, score=0.593 total time=  13.9s
[CV 5/5; 15/24] START criterion=entropy, max_depth=1, n_esti
mators=1000.........
[CV 5/5; 15/24] END criterion=entropy, max_depth=1, n_estima
tors=1000;, score=0.611 total time=  13.4s
[CV 2/5; 18/24] START criterion=entropy, max_depth=3, n_esti
mators=1000.........
[CV 2/5; 18/24] END criterion=entropy, max_depth=3, n_estima
tors=1000;, score=0.640 total time=  15.9s
[CV 1/5; 19/24] START criterion=entropy, max_depth=7, n_esti
mators=100..........
[CV 1/5; 19/24] END criterion=entropy, max_depth=7, n_estima
```

```
tors=100;, score=0.615 total time=   1.4s
[CV 2/5; 19/24] START criterion=entropy, max_depth=7, n_esti
mators=100.........
[CV 2/5; 19/24] END criterion=entropy, max_depth=7, n_estima
tors=100;, score=0.647 total time=   1.5s
[CV 3/5; 19/24] START criterion=entropy, max_depth=7, n_esti
mators=100.........
[CV 3/5; 19/24] END criterion=entropy, max_depth=7, n_estima
tors=100;, score=0.629 total time=   1.8s
[CV 4/5; 19/24] START criterion=entropy, max_depth=7, n_esti
mators=100.........
[CV 4/5; 19/24] END criterion=entropy, max_depth=7, n_estima
tors=100;, score=0.644 total time=   2.0s
[CV 2/5; 20/24] START criterion=entropy, max_depth=7, n_esti
mators=500.........
[CV 2/5; 20/24] END criterion=entropy, max_depth=7, n_estima
tors=500;, score=0.642 total time=   9.3s
[CV 1/5; 21/24] START criterion=entropy, max_depth=7, n_esti
mators=1000........
[CV 1/5; 21/24] END criterion=entropy, max_depth=7, n_estima
tors=1000;, score=0.619 total time=  20.3s
[CV 5/5; 21/24] START criterion=entropy, max_depth=7, n_esti
mators=1000........
[CV 5/5; 21/24] END criterion=entropy, max_depth=7, n_estima
tors=1000;, score=0.648 total time=  21.9s
[CV 4/5; 23/24] START criterion=entropy, max_depth=10, n_est
imators=500........
[CV 4/5; 23/24] END criterion=entropy, max_depth=10, n_estim
ators=500;, score=0.653 total time=  12.0s
[CV 3/5; 24/24] START criterion=entropy, max_depth=10, n_est
imators=1000........
[CV 3/5; 24/24] END criterion=entropy, max_depth=10, n_estim
ators=1000;, score=0.629 total time=  23.9s
[CV 2/5; 1/24] START criterion=gini, max_depth=1, n_estimato
rs=100.............
[CV 2/5; 1/24] END criterion=gini, max_depth=1, n_estimators
=100;, score=0.602 total time=   1.2s
[CV 5/5; 1/24] START criterion=gini, max_depth=1, n_estimato
rs=100.............
[CV 5/5; 1/24] END criterion=gini, max_depth=1, n_estimators
=100;, score=0.611 total time=   1.1s
[CV 4/5; 2/24] START criterion=gini, max_depth=1, n_estimato
rs=500.............
[CV 4/5; 2/24] END criterion=gini, max_depth=1, n_estimators
=500;, score=0.630 total time=   6.5s
[CV 3/5; 3/24] START criterion=gini, max_depth=1, n_estimato
rs=1000............
[CV 3/5; 3/24] END criterion=gini, max_depth=1, n_estimators
```

```
=1000;, score=0.605 total time=  12.3s
[CV 3/5; 4/24] START criterion=gini, max_depth=3, n_estimato
rs=100.............
[CV 3/5; 4/24] END criterion=gini, max_depth=3, n_estimators
=100;, score=0.625 total time=   1.0s
[CV 5/5; 4/24] START criterion=gini, max_depth=3, n_estimato
rs=100.............
[CV 5/5; 4/24] END criterion=gini, max_depth=3, n_estimators
=100;, score=0.637 total time=   1.1s
[CV 2/5; 5/24] START criterion=gini, max_depth=3, n_estimato
rs=500.............
[CV 2/5; 5/24] END criterion=gini, max_depth=3, n_estimators
=500;, score=0.634 total time=   7.1s
[CV 5/5; 5/24] START criterion=gini, max_depth=3, n_estimato
rs=500.............
[CV 5/5; 5/24] END criterion=gini, max_depth=3, n_estimators
=500;, score=0.624 total time=   6.1s
[CV 4/5; 6/24] START criterion=gini, max_depth=3, n_estimato
rs=1000............
[CV 4/5; 6/24] END criterion=gini, max_depth=3, n_estimators
=1000;, score=0.645 total time=  14.0s
[CV 4/5; 7/24] START criterion=gini, max_depth=7, n_estimato
rs=100.............
[CV 4/5; 7/24] END criterion=gini, max_depth=7, n_estimators
=100;, score=0.644 total time=   1.8s
[CV 3/5; 8/24] START criterion=gini, max_depth=7, n_estimato
rs=500.............
[CV 3/5; 8/24] END criterion=gini, max_depth=7, n_estimators
=500;, score=0.627 total time=   9.5s
[CV 1/5; 9/24] START criterion=gini, max_depth=7, n_estimato
rs=1000............
[CV 1/5; 9/24] END criterion=gini, max_depth=7, n_estimators
=1000;, score=0.618 total time=  17.9s
[CV 5/5; 9/24] START criterion=gini, max_depth=7, n_estimato
rs=1000............
[CV 5/5; 9/24] END criterion=gini, max_depth=7, n_estimators
=1000;, score=0.645 total time=  21.2s
[CV 4/5; 11/24] START criterion=gini, max_depth=10, n_estima
tors=500...........
[CV 4/5; 11/24] END criterion=gini, max_depth=10, n_estimato
rs=500;, score=0.650 total time=  11.7s
[CV 3/5; 12/24] START criterion=gini, max_depth=10, n_estima
tors=1000..........
[CV 3/5; 12/24] END criterion=gini, max_depth=10, n_estimato
rs=1000;, score=0.628 total time=  24.7s
[CV 2/5; 14/24] START criterion=entropy, max_depth=1, n_esti
mators=500.........
[CV 2/5; 14/24] END criterion=entropy, max_depth=1, n_estima
```

```
tors=500;, score=0.628 total time=   6.8s
[CV 5/5; 14/24] START criterion=entropy, max_depth=1, n_esti
mators=500.........
[CV 5/5; 14/24] END criterion=entropy, max_depth=1, n_estima
tors=500;, score=0.611 total time=   7.3s
[CV 3/5; 15/24] START criterion=entropy, max_depth=1, n_esti
mators=1000.........
[CV 3/5; 15/24] END criterion=entropy, max_depth=1, n_estima
tors=1000;, score=0.599 total time=  12.4s
[CV 3/5; 16/24] START criterion=entropy, max_depth=3, n_esti
mators=100.........
[CV 3/5; 16/24] END criterion=entropy, max_depth=3, n_estima
tors=100;, score=0.633 total time=   1.0s
[CV 4/5; 16/24] START criterion=entropy, max_depth=3, n_esti
mators=100.........
[CV 4/5; 16/24] END criterion=entropy, max_depth=3, n_estima
tors=100;, score=0.646 total time=   1.5s
[CV 2/5; 17/24] START criterion=entropy, max_depth=3, n_esti
mators=500.........
[CV 2/5; 17/24] END criterion=entropy, max_depth=3, n_estima
tors=500;, score=0.640 total time=   6.9s
[CV 4/5; 17/24] START criterion=entropy, max_depth=3, n_esti
mators=500.........
[CV 4/5; 17/24] END criterion=entropy, max_depth=3, n_estima
tors=500;, score=0.644 total time=   7.3s
[CV 4/5; 18/24] START criterion=entropy, max_depth=3, n_esti
mators=1000.........
[CV 4/5; 18/24] END criterion=entropy, max_depth=3, n_estima
tors=1000;, score=0.647 total time=  15.5s
[CV 1/5; 20/24] START criterion=entropy, max_depth=7, n_esti
mators=500.........
[CV 1/5; 20/24] END criterion=entropy, max_depth=7, n_estima
tors=500;, score=0.619 total time=   9.6s
[CV 5/5; 20/24] START criterion=entropy, max_depth=7, n_esti
mators=500.........
[CV 5/5; 20/24] END criterion=entropy, max_depth=7, n_estima
tors=500;, score=0.645 total time=   9.8s
[CV 4/5; 21/24] START criterion=entropy, max_depth=7, n_esti
mators=1000.........
[CV 4/5; 21/24] END criterion=entropy, max_depth=7, n_estima
tors=1000;, score=0.647 total time=  20.7s
[CV 1/5; 23/24] START criterion=entropy, max_depth=10, n_est
imators=500.........
[CV 1/5; 23/24] END criterion=entropy, max_depth=10, n_estim
ators=500;, score=0.618 total time=  13.3s
[CV 5/5; 23/24] START criterion=entropy, max_depth=10, n_est
imators=500.........
[CV 5/5; 23/24] END criterion=entropy, max_depth=10, n_estim
```

```
ators=500;, score=0.643 total time=  11.9s
[CV 4/5; 24/24] START criterion=entropy, max_depth=10, n_est
imators=1000.....
[CV 4/5; 24/24] END criterion=entropy, max_depth=10, n_estim
ators=1000;, score=0.650 total time=  23.8s
[CV 4/5; 1/24] START criterion=gini, max_depth=1, n_estimato
rs=100.............
[CV 4/5; 1/24] END criterion=gini, max_depth=1, n_estimators
=100;, score=0.642 total time=   1.7s
[CV 2/5; 2/24] START criterion=gini, max_depth=1, n_estimato
rs=500.............
[CV 2/5; 2/24] END criterion=gini, max_depth=1, n_estimators
=500;, score=0.617 total time=   6.2s
[CV 1/5; 3/24] START criterion=gini, max_depth=1, n_estimato
rs=1000............
[CV 1/5; 3/24] END criterion=gini, max_depth=1, n_estimators
=1000;, score=0.596 total time=  11.8s
[CV 1/5; 4/24] START criterion=gini, max_depth=3, n_estimato
rs=100.............
[CV 1/5; 4/24] END criterion=gini, max_depth=3, n_estimators
=100;, score=0.602 total time=   1.1s
[CV 2/5; 4/24] START criterion=gini, max_depth=3, n_estimato
rs=100.............
[CV 2/5; 4/24] END criterion=gini, max_depth=3, n_estimators
=100;, score=0.630 total time=   1.1s
[CV 4/5; 4/24] START criterion=gini, max_depth=3, n_estimato
rs=100.............
[CV 4/5; 4/24] END criterion=gini, max_depth=3, n_estimators
=100;, score=0.643 total time=   1.2s
[CV 1/5; 5/24] START criterion=gini, max_depth=3, n_estimato
rs=500.............
[CV 1/5; 5/24] END criterion=gini, max_depth=3, n_estimators
=500;, score=0.610 total time=   7.1s
[CV 4/5; 5/24] START criterion=gini, max_depth=3, n_estimato
rs=500.............
[CV 4/5; 5/24] END criterion=gini, max_depth=3, n_estimators
=500;, score=0.645 total time=   6.2s
[CV 3/5; 6/24] START criterion=gini, max_depth=3, n_estimato
rs=1000............
[CV 3/5; 6/24] END criterion=gini, max_depth=3, n_estimators
=1000;, score=0.628 total time=  14.1s
[CV 5/5; 7/24] START criterion=gini, max_depth=7, n_estimato
rs=100.............
[CV 5/5; 7/24] END criterion=gini, max_depth=7, n_estimators
=100;, score=0.641 total time=   1.8s
[CV 2/5; 8/24] START criterion=gini, max_depth=7, n_estimato
rs=500.............
[CV 2/5; 8/24] END criterion=gini, max_depth=7, n_estimators
```

```
=500;, score=0.644 total time=   9.7s
[CV 2/5; 9/24] START criterion=gini, max_depth=7, n_estimato
rs=1000.............
[CV 2/5; 9/24] END criterion=gini, max_depth=7, n_estimators
=1000;, score=0.642 total time=  18.2s
[CV 1/5; 10/24] START criterion=gini, max_depth=10, n_estima
tors=100............
[CV 1/5; 10/24] END criterion=gini, max_depth=10, n_estimato
rs=100;, score=0.614 total time=   2.0s
[CV 2/5; 10/24] START criterion=gini, max_depth=10, n_estima
tors=100............
[CV 2/5; 10/24] END criterion=gini, max_depth=10, n_estimato
rs=100;, score=0.643 total time=   2.1s
[CV 3/5; 10/24] START criterion=gini, max_depth=10, n_estima
tors=100............
[CV 3/5; 10/24] END criterion=gini, max_depth=10, n_estimato
rs=100;, score=0.622 total time=   3.4s
[CV 4/5; 10/24] START criterion=gini, max_depth=10, n_estima
tors=100............
[CV 4/5; 10/24] END criterion=gini, max_depth=10, n_estimato
rs=100;, score=0.650 total time=   2.9s
[CV 5/5; 10/24] START criterion=gini, max_depth=10, n_estima
tors=100............
[CV 5/5; 10/24] END criterion=gini, max_depth=10, n_estimato
rs=100;, score=0.635 total time=   2.4s
[CV 3/5; 11/24] START criterion=gini, max_depth=10, n_estima
tors=500............
[CV 3/5; 11/24] END criterion=gini, max_depth=10, n_estimato
rs=500;, score=0.631 total time=  12.1s
[CV 2/5; 12/24] START criterion=gini, max_depth=10, n_estima
tors=1000...........
[CV 2/5; 12/24] END criterion=gini, max_depth=10, n_estimato
rs=1000;, score=0.638 total time=  24.4s
[CV 1/5; 13/24] START criterion=entropy, max_depth=1, n_esti
mators=100..........
[CV 1/5; 13/24] END criterion=entropy, max_depth=1, n_estima
tors=100;, score=0.581 total time=   1.0s
[CV 2/5; 13/24] START criterion=entropy, max_depth=1, n_esti
mators=100..........
[CV 2/5; 13/24] END criterion=entropy, max_depth=1, n_estima
tors=100;, score=0.633 total time=   1.1s
[CV 3/5; 13/24] START criterion=entropy, max_depth=1, n_esti
mators=100..........
[CV 3/5; 13/24] END criterion=entropy, max_depth=1, n_estima
tors=100;, score=0.597 total time=   1.4s
[CV 4/5; 13/24] START criterion=entropy, max_depth=1, n_esti
mators=100..........
[CV 4/5; 13/24] END criterion=entropy, max_depth=1, n_estima
```

```
tors=100;, score=0.624 total time=   1.5s
[CV 5/5; 13/24] START criterion=entropy, max_depth=1, n_esti
mators=100.....
[CV 5/5; 13/24] END criterion=entropy, max_depth=1, n_estima
tors=100;, score=0.609 total time=   1.3s
[CV 1/5; 14/24] START criterion=entropy, max_depth=1, n_esti
mators=500.....
[CV 1/5; 14/24] END criterion=entropy, max_depth=1, n_estima
tors=500;, score=0.585 total time=   6.7s
[CV 4/5; 14/24] START criterion=entropy, max_depth=1, n_esti
mators=500.....
[CV 4/5; 14/24] END criterion=entropy, max_depth=1, n_estima
tors=500;, score=0.631 total time=   7.4s
[CV 2/5; 15/24] START criterion=entropy, max_depth=1, n_esti
mators=1000.....
[CV 2/5; 15/24] END criterion=entropy, max_depth=1, n_estima
tors=1000;, score=0.620 total time=  12.4s
[CV 1/5; 16/24] START criterion=entropy, max_depth=3, n_esti
mators=100.....
[CV 1/5; 16/24] END criterion=entropy, max_depth=3, n_estima
tors=100;, score=0.607 total time=   1.1s
[CV 2/5; 16/24] START criterion=entropy, max_depth=3, n_esti
mators=100.....
[CV 2/5; 16/24] END criterion=entropy, max_depth=3, n_estima
tors=100;, score=0.643 total time=   1.1s
[CV 5/5; 16/24] START criterion=entropy, max_depth=3, n_esti
mators=100.....
[CV 5/5; 16/24] END criterion=entropy, max_depth=3, n_estima
tors=100;, score=0.615 total time=   1.4s
[CV 1/5; 17/24] START criterion=entropy, max_depth=3, n_esti
mators=500.....
[CV 1/5; 17/24] END criterion=entropy, max_depth=3, n_estima
tors=500;, score=0.605 total time=   7.0s
[CV 5/5; 17/24] START criterion=entropy, max_depth=3, n_esti
mators=500.....
[CV 5/5; 17/24] END criterion=entropy, max_depth=3, n_estima
tors=500;, score=0.622 total time=   7.2s
[CV 3/5; 18/24] START criterion=entropy, max_depth=3, n_esti
mators=1000.....
[CV 3/5; 18/24] END criterion=entropy, max_depth=3, n_estima
tors=1000;, score=0.628 total time=  15.5s
[CV 5/5; 19/24] START criterion=entropy, max_depth=7, n_esti
mators=100.....
[CV 5/5; 19/24] END criterion=entropy, max_depth=7, n_estima
tors=100;, score=0.641 total time=   1.7s
[CV 3/5; 20/24] START criterion=entropy, max_depth=7, n_esti
mators=500.....
[CV 3/5; 20/24] END criterion=entropy, max_depth=7, n_estima
```

```
tors=500;, score=0.627 total time=   9.4s
[CV 2/5; 21/24] START criterion=entropy, max_depth=7, n_esti
mators=1000.........
[CV 2/5; 21/24] END criterion=entropy, max_depth=7, n_estima
tors=1000;, score=0.645 total time=  20.2s
[CV 1/5; 22/24] START criterion=entropy, max_depth=10, n_est
imators=100.........
[CV 1/5; 22/24] END criterion=entropy, max_depth=10, n_estim
ators=100;, score=0.616 total time=   2.1s
[CV 2/5; 22/24] START criterion=entropy, max_depth=10, n_est
imators=100.........
[CV 2/5; 22/24] END criterion=entropy, max_depth=10, n_estim
ators=100;, score=0.638 total time=   1.9s
[CV 3/5; 22/24] START criterion=entropy, max_depth=10, n_est
imators=100.........
[CV 3/5; 22/24] END criterion=entropy, max_depth=10, n_estim
ators=100;, score=0.624 total time=   3.0s
[CV 4/5; 22/24] START criterion=entropy, max_depth=10, n_est
imators=100.........
[CV 4/5; 22/24] END criterion=entropy, max_depth=10, n_estim
ators=100;, score=0.652 total time=   2.6s
[CV 2/5; 23/24] START criterion=entropy, max_depth=10, n_est
imators=500.........
[CV 2/5; 23/24] END criterion=entropy, max_depth=10, n_estim
ators=500;, score=0.641 total time=  13.4s
[CV 1/5; 24/24] START criterion=entropy, max_depth=10, n_est
imators=1000........
[CV 1/5; 24/24] END criterion=entropy, max_depth=10, n_estim
ators=1000;, score=0.616 total time=  24.6s
[CV 5/5; 24/24] START criterion=entropy, max_depth=10, n_est
imators=1000........
[CV 5/5; 24/24] END criterion=entropy, max_depth=10, n_estim
ators=1000;, score=0.644 total time=  16.3s
```

In [9…

```
model5 = RandomForestClassifier(max_depth =7, n_jobs= -1, min

Y_pred_test = model_fit_predict(model2, X_train, Y_train, X_t

f1 = round(f1_score(Y_tests, Y_pred_test),2)

acc = round(accuracy_score(Y_tests, Y_pred_test),2)

recall=round(recall_score(Y_tests, Y_pred_test, pos_label=0),

pre = round(precision_score(Y_tests, Y_pred_test),2)

print(f"Accuracy, precision and f1-score for training data ar
```

```
Accuracy, precision and f1-score for training data are 0.61,
0.62 and 0.61 respectively
```